User's Guide

GoldSim

Probabilistic Simulation Environment

Volume 2 of 2

GoldSim TECHNOLOGY GROUP

Version 11.1 (May 2014)

The GoldSim User's Guide is divided into two volumes. Chapters 1 through 7 are included in Volume 1. Chapters 8 through 10, along with the appendices, are included in Volume 2. The full Table of Contents, Index and Glossary are provided in both volumes.

GoldSim Technology Group
22500 SE 64th Place, Suite 240
Issaquah, Washington 98027 USA

Visit us at our web site: www.goldsim.com
Email us at: software@goldsim.com

Contents

Chapter 10: Advanced Modeling Concepts 725

Chapter 8: Displaying Results in GoldSim

Often the most effective way to describe, explore, and summarize a set of numbers - even a very large set - is to look at pictures of those numbers.

Edward Tufte, *The Visual Display of Quantitative Information*

Chapter Overview

GoldSim has powerful charting and display functions that allow you to view your simulation results in a variety of ways. You can plot time histories of your data, view probability distributions, create scatter plots and bar charts, and view tables of results. You can also combine multiple results on a single plot, and view multiple plots simultaneously. To facilitate the creation of report quality graphics, you can modify and save *chart styles*, which allow you to customize (and reuse) the style (i.e., appearance) for each type of chart.

This chapter describes how you can create and view the results of your simulations.

In this Chapter

The following topics are discussed in this chapter:

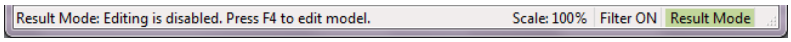
- Displaying Results: An Overview
- Viewing Time History Results
- Viewing Distribution Results
- Viewing Multi-Variate Results
- Viewing Final Values for Arrays
- Editing the Appearance of a Chart
- Creating and Using Chart Styles
- Exporting Results
- Copying and Exporting Result Displays

Displaying Results: An Overview

The sections below provide the basic information you need to understand the types of result displays you can produce in GoldSim. They also provide references to topics that describe these various result display features in greater detail.

Understanding Result Mode

A GoldSim model file is always in one of four modes (or states): Edit Mode, Run Mode, Result Mode, or Scenario Mode. The mode that the file is in is indicated in the status bar at the bottom of the GoldSim window.



Read more: Understanding Simulation Modes (page 456).

Results can be viewed in two modes: Result Mode and Scenario Mode. Result Mode is the most common way that you will view results. Scenario Mode only exists if you are using GoldSim's scenario feature. Viewing results in Scenario Mode is similar to viewing results in Result Mode, with a number of key limitations.

Read more: Viewing Scenario Results (page 532).

After you run your model (e.g., by pressing **F5**), it will automatically be placed in Result Mode. When in Result Mode, elements with outputs that have been saved are identified in two ways: 1) the elements (and their outputs) are **bold** in the browser (and output interfaces); and 2) the element output ports in the graphics pane are green (rather than red).

When a model is in Result Mode, you can navigate it, open property dialogs, and view result plots. You can also move elements around within the graphics pane, add graphics (e.g., text, images), and add and edit Result elements.

You cannot, however, edit the model in any way that would change it's behavior (e.g., you cannot change values in property dialogs or add or delete elements).

This is indicated in the status bar, and by the cursor, which takes on a different appearance to indicate that editing is disabled. GoldSim enforces this rule to ensure that the results and inputs to a model do not become inconsistent.

If you press **F4**, choose **Run|Return to Edit Mode** from the file menu, or press the Edit Mode button in the standard toolbar, the results will be discarded and the model will be returned to Edit Mode.

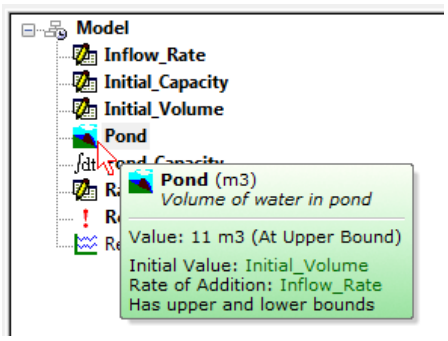
By default, the Run Controller is not visible when a model is in Result Mode (although you can optionally choose to show the Run Controller when in Result Mode).

Read more: Customizing the Behavior of the Run Controller (page 462).

Viewing "Last Value" Results in Tool-Tips

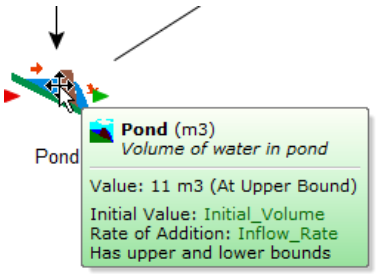
Although GoldSim provides a variety of powerful ways in which you can display simulation results in the form or charts and tables, the simplest way to view basic results is via tool-tips.

In Result Mode, an output's last Value (the value at the end of the last realization) will be displayed in a tool-tip when the cursor is held over it in a browser (or output interface).



Read more: Using Tool-Tips (page 113).

If the output is the primary output of an element, the last Value will also be shown in the tool-tip displayed when the cursor is held over the element itself.



Tool-tips are always displayed for scalar outputs. If you hold the cursor over a vector or a matrix, only a portion of the Last Values may be displayed (if the array is large). Note, however, that if you expand the array in a browser or output interface and hold the cursor over a single item, the last Value for that item will be displayed.

Read more: Using Vectors and Matrices (page 726).

Note: The last Value for an output is displayed in tool-tips regardless of whether results (Time Histories or Final Values) have been saved for the output.

Read more: Saving Outputs as Results (page 453).

Note: You can control the number of significant figures displayed in tool-tips from the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).

Read more: The Results Tab of the Options Dialog (page 409).

Viewing the Four Basic Result Types

GoldSim provides four ways in which you can display results in graphical and tabular form:

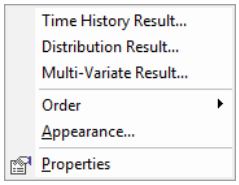
Time History Results: Values at selected times for a particular output.

Distribution Results: Probability distributions of the final value (i.e., the value at the end of each realization) of an uncertain output.

Multi-Variate Results: Comparisons between the final values of variables (outputs), such as scatter plots, correlation tables, and sensitivity analyses; and

Array Results: Final values for vector and matrix outputs.

Right-clicking on an element with a primary output (in the graphics pane or browser) or on a specific output (in an output interface or a browser) which has been saved will provide a context menu for displaying these four types of results.



Depending on the output type and what kinds of results have been saved, up to three of the result options will be available.

Note: Time History Results are only available if you have selected **Save Time Histories** for the output. Multi-Variate Results and Distribution Results are only available if you have selected **Save Final Values** for the output. Array Results are only available if the output is a vector or a matrix and you have selected **Save Final Values** for the output.

Read more: Saving Outputs as Results (page 453).

All four result types can be viewed either as a *chart* or a *table* (and in some cases, there are multiple types of charts and/or tables that can be viewed for a particular type of result).

A chart or table accessed and viewed in this manner is referred to as an ***interactive result***. For each type of interactive result, there is a default display that is shown (a chart for all but the Distribution Result, which shows a summary page with both chart and tabular information). Once a result is displayed, you can subsequently switch back and forth between different chart and table views.

Charts or tables can also be viewed using specialized elements referred to as ***Result elements***. Result elements allow you to organize all of your key result charts or tables, and allow you to access those results via a simple double-click.

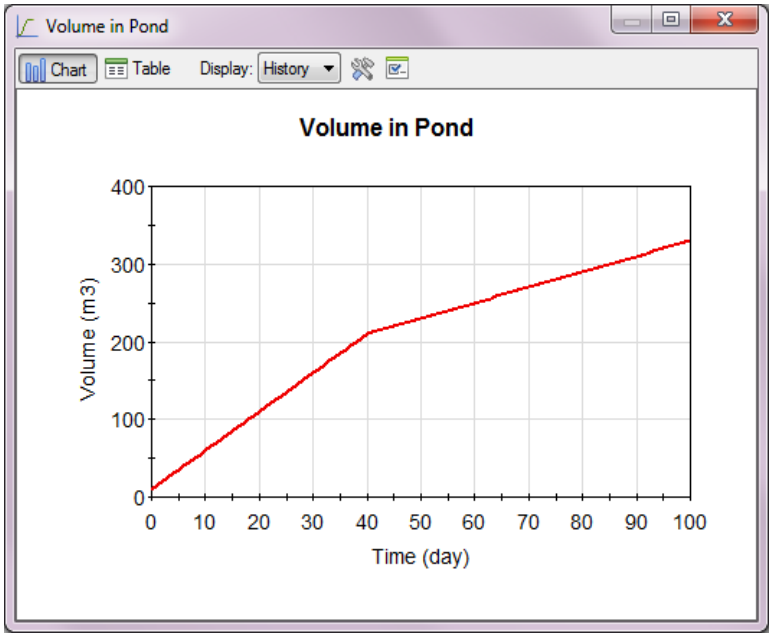
Read more: Creating and Using Result Elements (page 526).

Note: Interactive Time History Results for an output are not available for multi-realization runs unless the output is connected to a Result element.

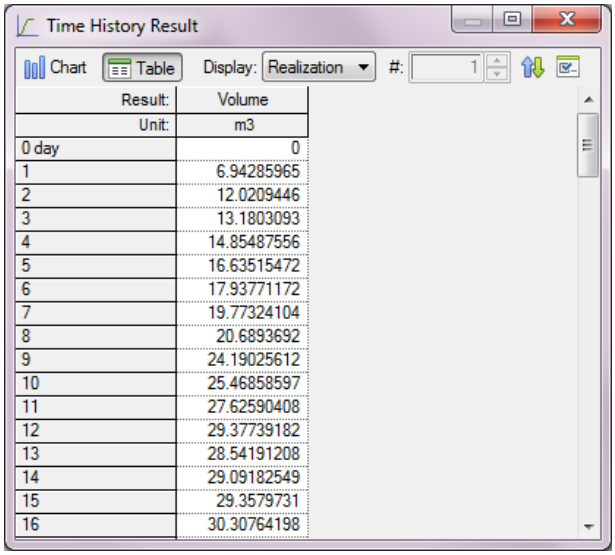
Examples of the four types of result displays are provided below. Details regarding the manner in which these result displays can be manipulated and customized are provided in subsequent sections.

Time History Results

Time History results show the "history" of a particular output as a function of time, and are probably the most common form of result display you will use. A time history chart looks like this:



The same information can also be viewed in table form, in which the time is shown in one column, and the output's value is shown in the other:

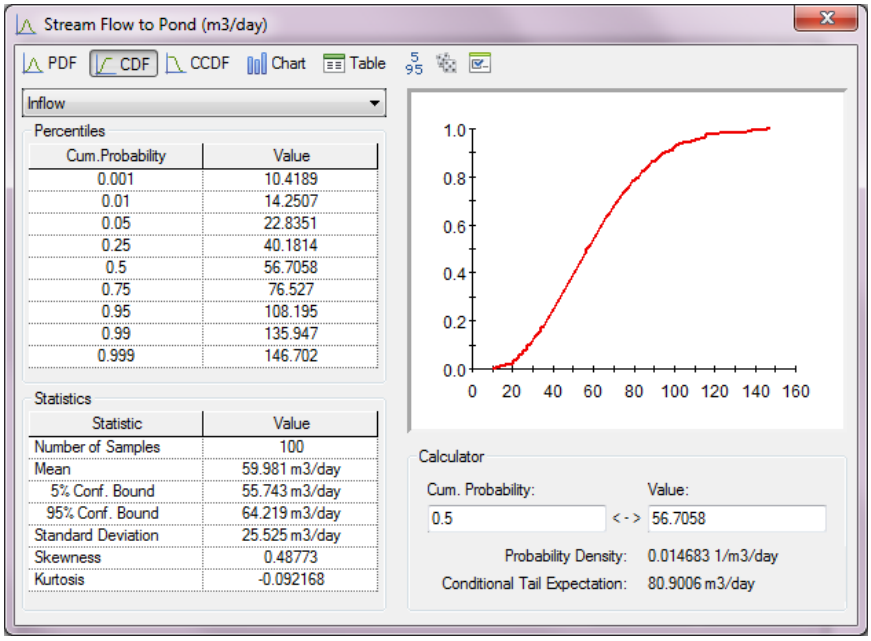


Result:	Volume
Unit:	m3
0 day	0
1	6.94285965
2	12.0209446
3	13.1803093
4	14.85487556
5	16.63515472
6	17.93771172
7	19.77324104
8	20.6893692
9	24.19025612
10	25.46858597
11	27.62590408
12	29.37739182
13	28.54191208
14	29.09182549
15	29.3579731
16	30.30764198

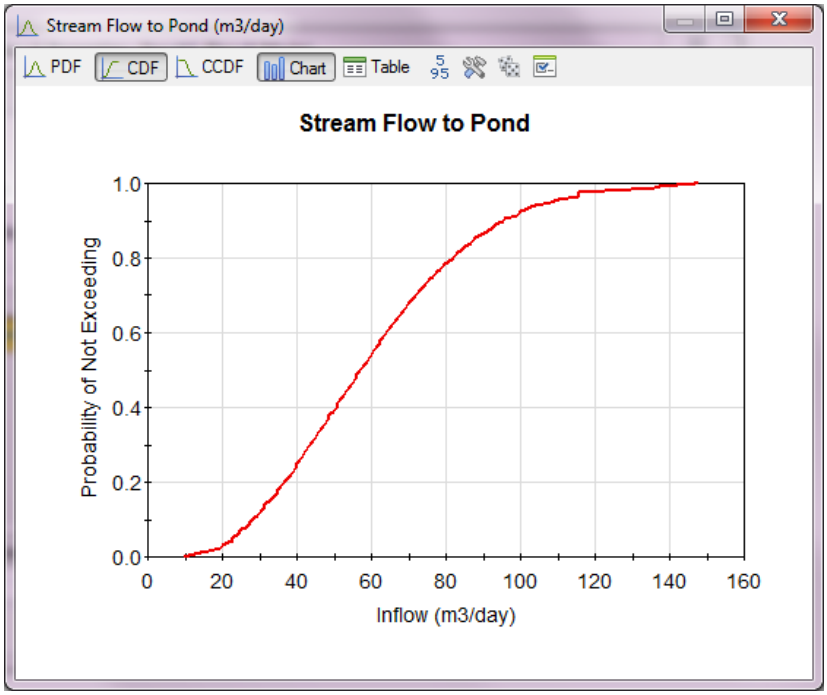
Distribution Results

In many systems, you may be uncertain about some of the input parameters. In such a case, GoldSim allows you to define these parameters as probability distributions. If at least one of the inputs to a model is a probability distribution, by definition, the outputs are probability distributions. (Basic concepts of probabilistic simulation are provided in Appendix A.) Distribution results provide a way to view the final values of uncertain (probabilistic) outputs.

The summary display for a distribution result (the default display for an interactive result) combines graphical and tabular information, and looks like this:

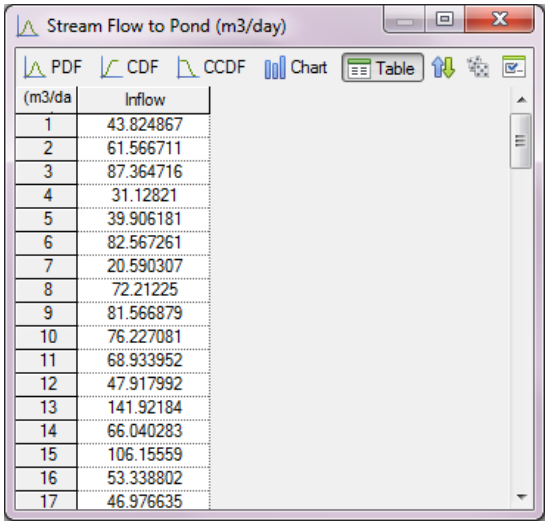


The chart display for a distribution expands the small chart provided in the upper right-hand corner of the summary display shown above.



Note that distribution charts can be viewed as probability density functions (PDFs), cumulative distribution functions (CDFs), or complementary cumulative distribution functions (CCDFs).

The table display for a distribution shows the final value for each realization:



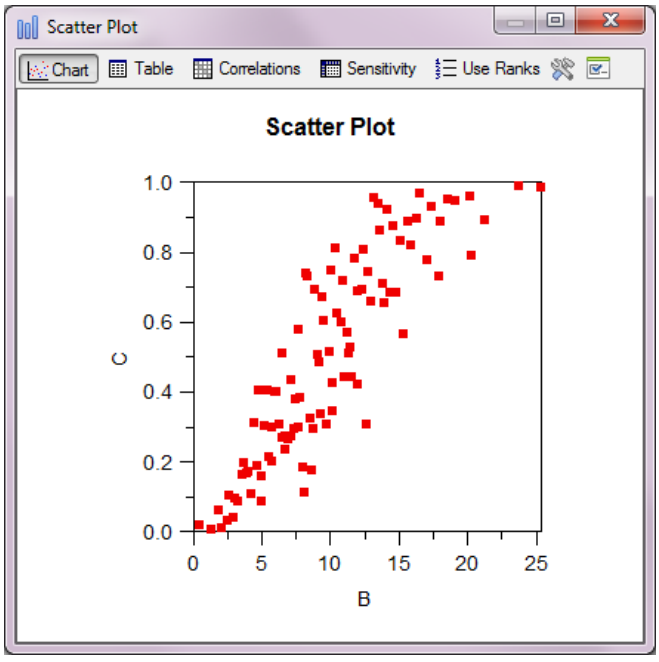
(m3/da	Inflow
1	43.824867
2	61.566711
3	87.364716
4	31.12821
5	39.906181
6	82.567261
7	20.590307
8	72.21225
9	81.566879
10	76.227081
11	68.933952
12	47.917992
13	141.92184
14	66.040283
15	106.15559
16	53.338802
17	46.976635

Multi-Variate Results

When evaluating the results of your simulations, you will sometimes want to view multi-variate results, in which multiple outputs are analyzed in graphical or tabular form.

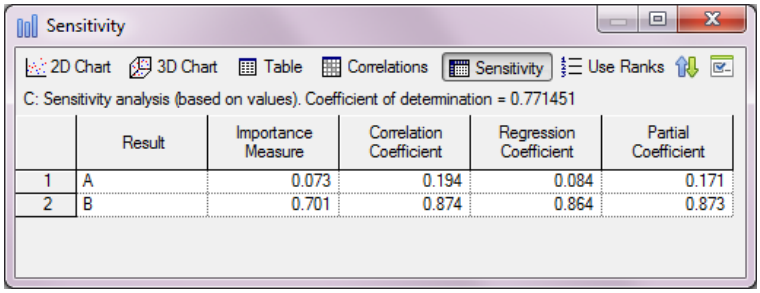
To view multi-variate results, you select a specific output that you are interested in, along with the input variables (which are typically Stochastics) that may have affected that result.

One example of this is the 2-D scatter plot, in which the final value of one output is plotted against the final value of another:



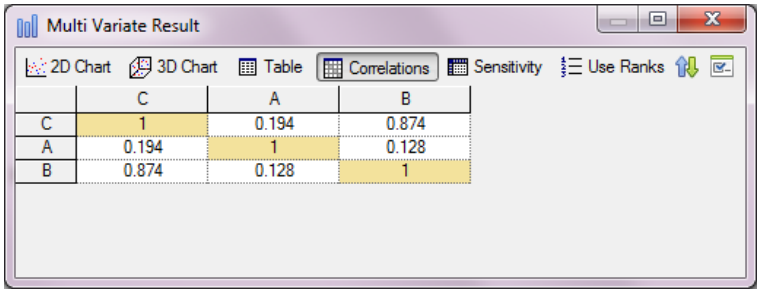
GoldSim also allows you to create a 3-D scatter plot, in which one output is plotted against two others.

GoldSim provides two very useful tabular multi-variate result outputs to support sensitivity and uncertainty analysis. One table provides a variety of statistical sensitivity analysis outputs (coefficient of determination, correlation coefficients, standardized regression coefficients, partial correlation coefficients, and importance measures):



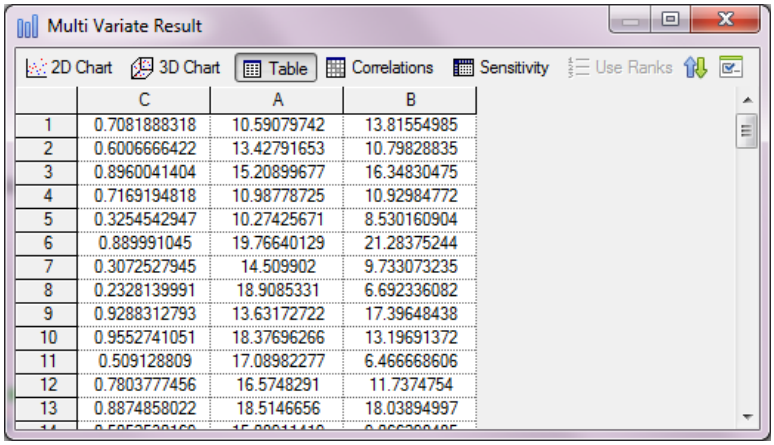
	Result	Importance Measure	Correlation Coefficient	Regression Coefficient	Partial Coefficient
1	A	0.073	0.194	0.084	0.171
2	B	0.701	0.874	0.864	0.873

A second table displays a complete correlation matrix, which indicates the degree to which all selected variables are correlated to each other:



	C	A	B
C	1	0.194	0.874
A	0.194	1	0.128
B	0.874	0.128	1

You can also view the raw data display of multiple variables, showing the final values for each realization for each of a number of selected outputs:



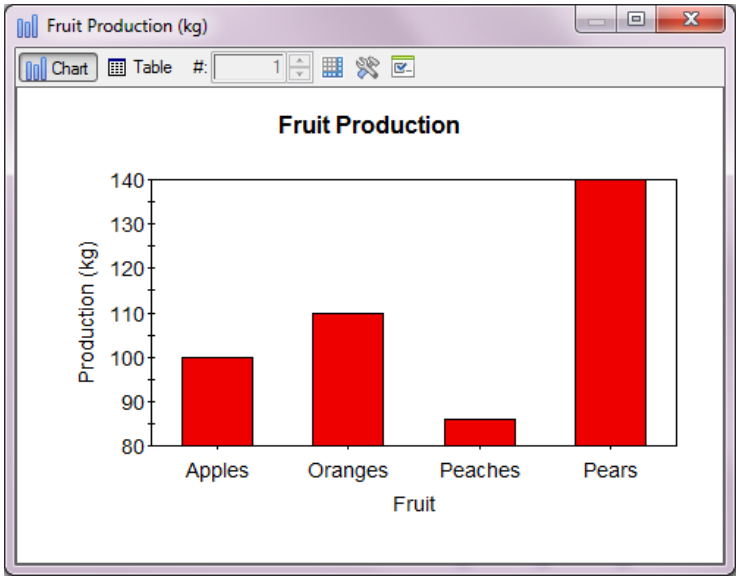
	C	A	B
1	0.7081888318	10.59079742	13.81554985
2	0.6006666422	13.42791653	10.79828835
3	0.8960041404	15.20899677	16.34830475
4	0.7169194818	10.98778725	10.92984772
5	0.3254542947	10.27425671	8.530160904
6	0.889991045	19.76640129	21.28375244
7	0.3072527945	14.509902	9.733073235
8	0.2328139991	18.9085331	6.692336082
9	0.9288312793	13.63172722	17.39648438
10	0.9552741051	18.37696266	13.19691372
11	0.509128809	17.08982277	6.466668606
12	0.7803777456	16.5748291	11.7374754
13	0.8874858022	18.5146656	18.03894997

Array Results

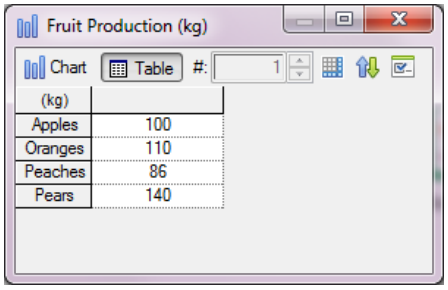
Many complex models will utilize vectors and matrices, collectively referred to as arrays. GoldSim allows you to view array results in both graphical and tabular form.

Read more: Using Vectors and Matrices (page 726).

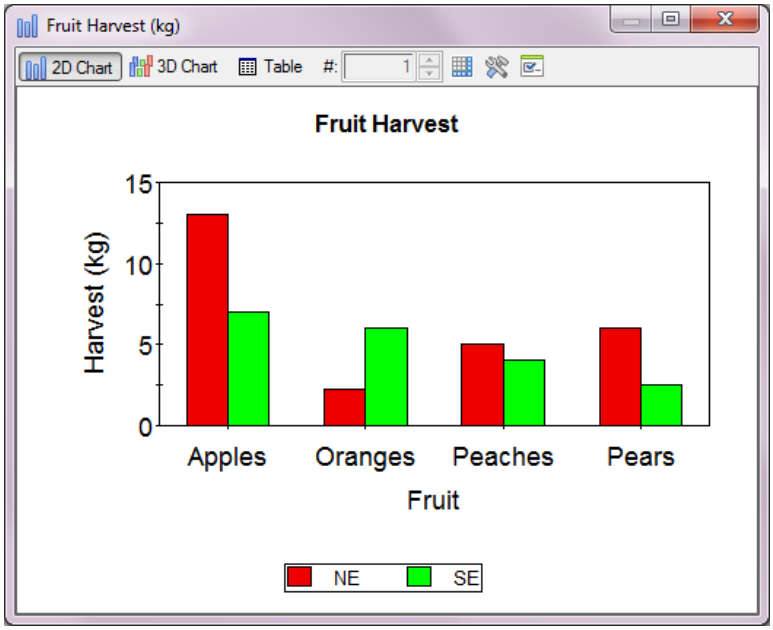
A chart display of a vector looks like this:



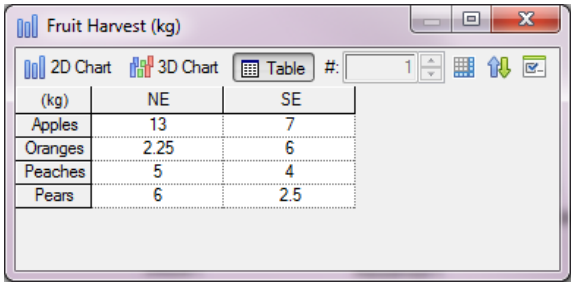
In tabular form, the vector looks like this:



One form of a chart display for a matrix looks like this:



In tabular form, the matrix is displayed as follows:



Using Result Display Windows

Although GoldSim allows you to view four different types of results (time histories, distributions, multi-variate results, and array results), the result display windows for these result types share a number of common characteristics:

- All four results can be viewed either as a *chart* or a *table* (and in some cases, there are multiple types of charts and/or tables that can be viewed for a particular type of result). For interactive results, the default is to show a chart (except for the Distribution Result, which shows a summary page with both chart and tabular information). Once a result is displayed, you can subsequently switch back and forth between different chart and table views.

- Although you can resize and maximize an interactive chart or table result display window, it is always *modal*. That is, it cannot be minimized, and with the exception of viewing the result properties dialog (which can be displayed simultaneously and share the focus with the result display window), it retains the focus while it is displayed. As a result, you must close the result display window before you can edit any other part of your GoldSim model.

  Read more: Viewing and Editing Result Properties (page 520).

- GoldSim does, however, provide a mechanism (Result elements) by which you can display one or more *modeless* result display windows (which allow you to navigate and edit other parts of the GoldSim model while keeping the windows open).

  Read more: Creating and Using Result Elements (page 526).

- Chart and table result display windows for all four types of results have a common set of buttons which provide access to specialized dialogs and/or allow you to carry out specific actions.

- You can control the number of significant figures displayed in result displays from the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).

  Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 524).

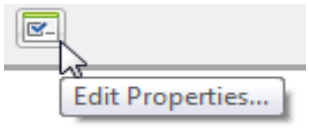
- Charts for all four result types provide a number of common functions, such as context-sensitive menus, data tips and the ability to zoom in on a section of the chart.

  Read more: Using Context Menus in Charts (page 522); Viewing Data Tips in Charts (page 523); Zooming in on a Chart (page 523).

Viewing and Editing Result Properties

For each type of interactive result, there is a default display that is shown (a chart for all but the Distribution Result, which shows a summary page with both chart and tabular information). Once a result is displayed, you can subsequently switch back and forth between different chart and table views.

Whenever you are viewing a result display, you can choose to view the *properties* for that particular result. All result display windows have a toolbar at the top. This toolbar contains buttons, a number of which are common to all result display windows. For all result display windows, the furthest button to the right is for displaying the Result Properties dialog:



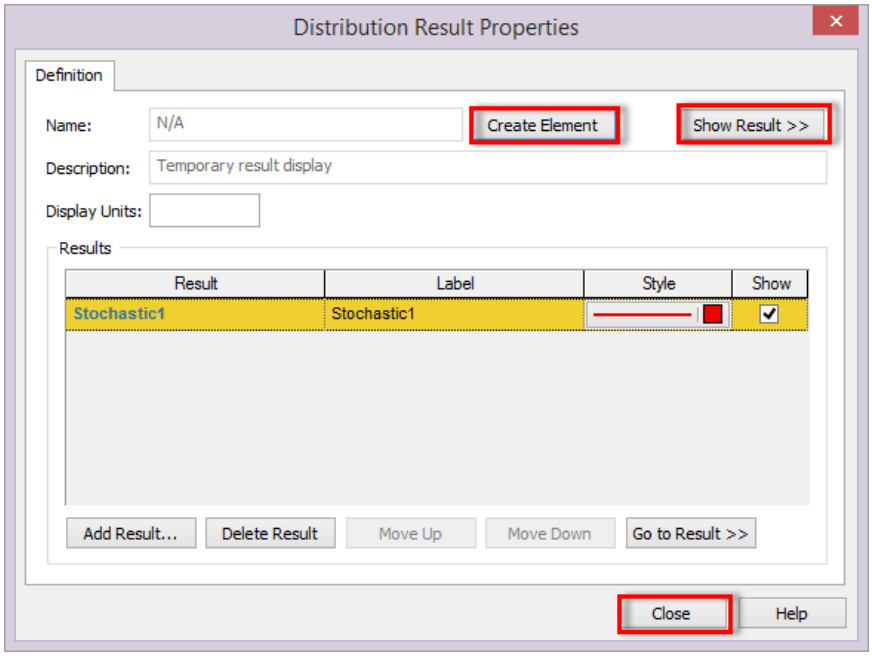
Pressing this button displays the Result Properties dialog for the result. Note that when you press this button, it does not close the result display window you were displaying. Both windows are displayed simultaneously (although you can subsequently close one or the other). When viewing results, any changes you make to the Result Properties are immediately represented in the result display.

The Result Properties dialog is always modal. That is, it cannot be minimized, and with the exception of viewing the result display (which can be displayed simultaneously and share the focus with the Properties dialog), it retains the focus while it is displayed. As a result, you must close the Result Properties dialog before you can edit any other part of your GoldSim model.

Result Properties differ somewhat for each of the four basic result types.

Read more: Viewing the Properties of a Time History Result (page 537); Viewing the Properties of a Distribution Result (page 597); Viewing the Properties of a Multi-Variate Result (page 633); Viewing the Properties of an Array Result (page 649).

The Properties for a Distribution Result are shown below, and the attributes that are shared among all result types are highlighted:



The following attributes are common to Result Properties dialogs:

- At the top of the page is a button to **Create Element**. By pressing this button, a Result element is created. When you do so, the key change that will be noticeable is that the Name will now be editable. The

Create Element button becomes an Appearance button (which is used to modify the appearance of the element itself).

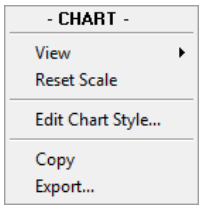
Read more: Creating and Using Result Elements (page 526).

- The **Show Result >>** button opens a result display window (while keeping the Properties dialog simultaneously open).
- The **Close** button closes the Properties dialog.

In addition, all Result Properties dialogs (except for Arrays) allow you to add additional results to the display (i.e., multiple outputs) using the **Add Result…** button, and jump directly to the selected result using the **Go to Result >>** button. You can also modify the **Label** for each result (which can subsequently be displayed in legends and headers, footers and axes labels).

Using Context Menus in Charts

All result charts in GoldSim have context menus from which you can carry out a variety of actions. By right-clicking anywhere in a chart, the following context menu is available:



The options in this menu are as follows:

View: Expands to show three options (Show Header, Show Footer, Show Legend). This allows you to toggle these on and off.

Reset Scale: GoldSim allows you to zoom within a chart (by dragging your cursor from one point of the chart to another *while holding down the Ctrl key*). Reset Scale allows you to reset to the original scale.

Read more: Zooming in on a Chart (page 523).

Edit Chart Style… : This provides access to a dialog for editing the chart style.

Read more: Editing the Appearance of a Chart (page 661).

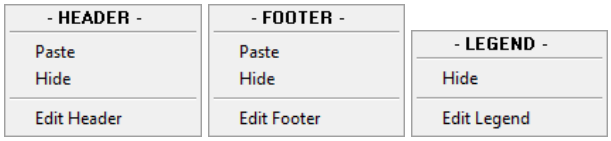
Copy: Copies the chart to the clipboard (from where it subsequently can be pasted into another application).

Read more: Copying a Chart or Table (page 691).

Export…: Exports the chart to a selected graphics format.

Read more: Exporting a Chart (page 691).

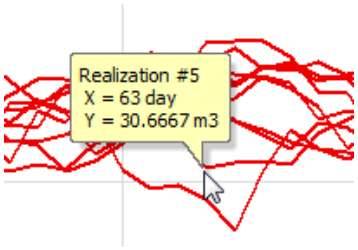
In addition to the Chart context menu, by right-clicking on a specific item (the Header, Footer or Legend), you can access a context menu that pertains just to that item:



These menus can be used to Hide the item, edit it directly (by going directly to the appropriate tab on the Chart Style dialog), or, in the case of Headers and Footers, paste text.

Viewing Data Tips in Charts

GoldSim allows you to view a "data tip" by holding the cursor over a data point in a chart. The example below shows a data tip for a point in a time history chart:



In this particular case, the data tip displays the realization number for the curve, and X and Y values of the data point. (For a time history chart, the X axis is always time).

Zooming in on a Chart

In some situations, you may want to zoom in on a portion of a chart that is of particular interest. You can, of course, edit the style of the chart and change the range of the axes which are displayed.

Read more: Editing the Appearance of a Chart (page 661).

GoldSim also provides a more direct way to zoom in on a section of a chart. In particular, if you drag your cursor from one point of the chart to another *while holding down the Ctrl key* you will create a selection box.

Releasing the left mouse button zooms in on the selected portion of the chart. To return to the original scale, right-click on the chart, and select **Reset Scale** from the chart context menu, or press **Shft+R**.

Selecting Items and Copying Values in Result Tables

In some cases, you may want to select, copy and paste values from a result table into another application (e.g., a spreadsheet).

Selecting rows or columns in a result table is straightforward. You can select a row or column by left-clicking in any row or column label:

Result1

Chart | Table | Display: All Realizations

Result:	Pond	Pond	Pond	Pond
Unit:	m3	m3	m3	m3
Displaying:	#1	#2	#3	#4
0 day	0	0	0	0
1	4.206027031	3.577036142	5.109113216	5.734968185
2	8.455780983	7.150896549	8.285830498	9.349498749
3	11.51830292	10.3004427	11.95729256	12.70890427
4	13.43139648	13.25799179	14.37914181	15.93443688
5	16.82522583	17.06479073	16.077425	17.36985016
6	19.32510567	19.98107529	16.27394485	19.57420921
7	21.09776115	21.45735931	16.787117	21.87093353
8	24.10535622	23.91679382	18.49937248	25.49660683
9	24.39035416	24.22426414	21.40023994	25.98937798
10	25.8646965	25.4542675	23.38226128	26.63475037
11	25.97337532	27.05109406	24.19250679	27.82192421
12	27.73983002	26.78435898	23.51561737	28.10332485
13	29.05348587	26.87271881	25.44615364	31.01965141
14	30.56312752	28.77239609	26.12899017	30.31923866
15	32.45275497	29.94305992	27.97798347	31.64370918
16	30.66819382	31.37153244	28.96376991	31.03497314
17	32.05984879	31.96710205	30.66625977	30.17387772
18	33.81079865	31.62252426	29.77319145	28.92910004
19	34.60662079	34.11064148	29.34663391	29.32118225
20	34.81547928	34.50789642	30.18501282	31.49066925

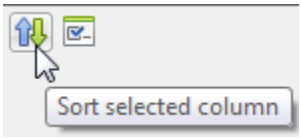
Contiguous rows or columns can be selected by dragging the mouse while it is pressed. Non-contiguous rows or columns can be selected by holding the Ctrl

key down while left-clicking. The entire table can be selected by clicking on the cell in the upper left-hand corner of the table.

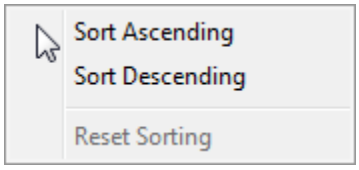
Once a selection has been made, it can be copied to the clipboard by pressing **Ctrl+C**.

Sorting Values in Result Tables

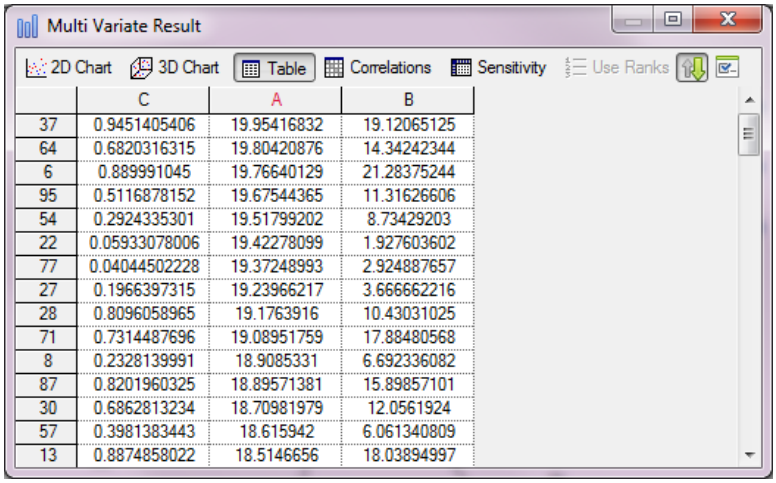
All result tables can be sorted in ascending or descending order. To sort a table, simply select a column (by clicking in it) and press the Sort button:



When you do so, a context menu will be displayed allowing you to sort in ascending or descending order:



After selecting one of these options, that column then becomes the sort key for the table (all columns are sorted based on that column). Whenever the data is sorted, the header of the sort key is shown in red:



	C	A	B
37	0.9451405406	19.95416832	19.12065125
64	0.6820316315	19.80420876	14.34242344
6	0.889991045	19.76640129	21.28375244
95	0.5116878152	19.67544365	11.31626606
54	0.2924335301	19.51799202	8.73429203
22	0.05933078006	19.42278099	1.927603602
77	0.04044502228	19.37248993	2.924887657
27	0.1966397315	19.23966217	3.666662216
28	0.8096058965	19.1763916	10.43031025
71	0.7314487696	19.08951759	17.88480568
8	0.2328139991	18.9085331	6.692336082
87	0.8201960325	18.89571381	15.89857101
30	0.6862813234	18.70981979	12.0561924
57	0.3981383443	18.615942	6.061340809
13	0.8874858022	18.5146656	18.03894997

In this case, the second column is the sort key, and the header is shown in red.

You can choose a new sort key by simply clicking on a different column and pressing the **Sort** button again.

Note that the **Sort** button changes its appearance when sorting is on. The button appears depressed, and one of the two arrows appears larger, and points in the direction of decreasing values (i.e., if the sort is in descending order the larger arrow points down; if it is in ascending order, the larger arrow points up).

To turn sorting off, press the **Sort** button again and select **Reset Sorting**.

Controlling Significant Figures and Scientific Notation in Result Displays

GoldSim provides a number of options for controlling the way that Results are displayed. These include the minimum number of sigificant figures to use, whether or not scientific notation is used, how conditions are displayed and how currencies are displayed. These options are controlled via the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).

Read more: The Results Tab of the Options Dialog (page 409).

In addition, you can control the minimum number of signficant figures to use and the magnitude above which scientific notation is used if directly from chart and table displays using accelerator keys:

Mininum number of signficant figures to display: This setting can be changed dynamically from within most result displays using **Alt-Left** and **Alt-Right**. The default value is 4.

Use scientific notation if absolute value is >=: This setting can be changed dynamically from within most result displays using **Alt-Up** and **Alt-Down**. The default value is 6 ((and GoldSim will always use scientific notation if the magnitude of the value is less than 0.0001 or greater than 1e10).

Note: When labeling chart axes, GoldSim respects the specified scientific notation setting, but ignores the signficant figures setting (significant figures in chart axes are determined automatically and cannot be user-controlled).

Size of Values Displayed

Although internal calculations are carried out using double precision numbers, results are only stored as single precision numbers (in order to reduce storage requirements). As a result, result values can only be relied upon to a maximum of 7 significant figures. For date outputs (e.g., outputs of the Milestone element), dates displayed in and around 2000 are only accurate to approximately 2 minutes.

This also means that when results are viewed in tables or charts, the range of values that can be displayed is between -1.2E-38 and 3.4E38.

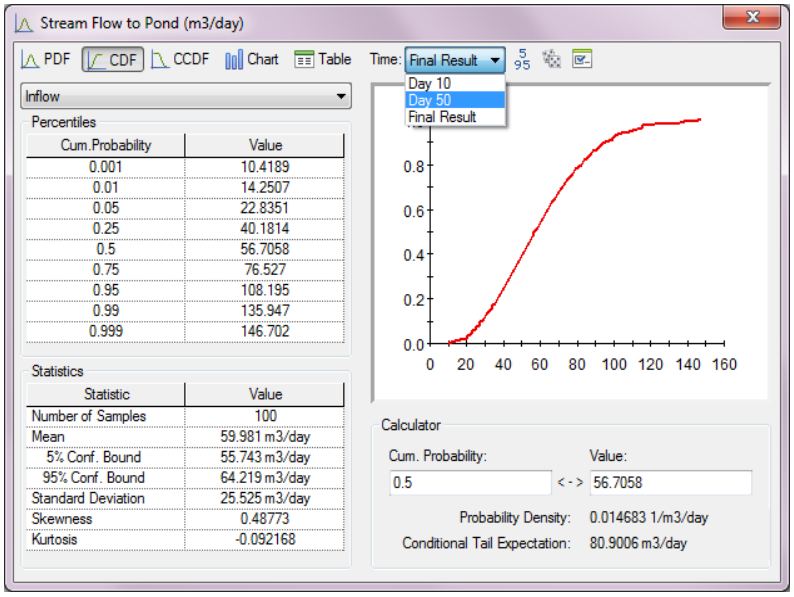
Viewing Results at Capture Points

Distributions, Multi-Variate results, and Array results all operate on Final Values. That is, they allow you to view the value at the end of each realization.

In some cases, however, you may also want to capture these results at other times in the simulation (rather than just the end of the simulation). This can be useful, for example, if you wanted to view a result distribution for an output at various times during a simulation. GoldSim facilitates this by allowing you to create Capture Points at which these results are also made available.

Read more: Creating Capture Points for Final Value Results (page 428).

If you have created Capture Points, an additional drop-list (labeled “Time”) is added to the result display windows for Distribution, Multi-Variate, and Array results:



To view the results for a particular Capture Point, you simply select it from the drop-list.

A simple example file illustrating Capture Points (CapturePoints.gsm) can be found in the Timestepping subfolder of the General Examples folder in your GoldSim directory.

Note: Several elements (Stochastics, SubModels, Spreadsheet elements) produce a special type of complex output representing all the distribution information (Distribution output). This output can be viewed in a Distribution Result. However, when doing so, Capture Points are ignored (when viewing a Distribution output, Capture Points don't apply, since a Distribution output contains a single "snapshot" and cannot represent multiple "snapshots").

Creating and Using Result Elements

While the interactive result display procedure (in which you right-click on an element or an output and select a result display option from a context menu) is appropriate in many cases, it has several limitations:

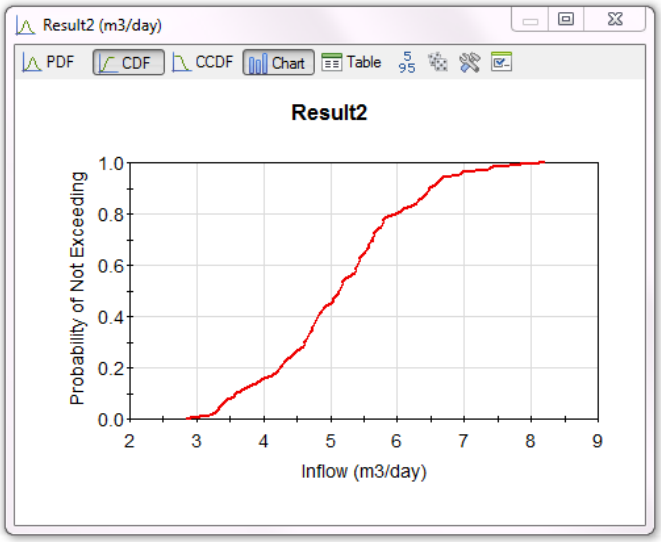
- It requires several steps to actually view the result (expand an element's outputs, right-click on an output, select display options).
- The outputs you want to display may be scattered across various portions of your model, forcing you to navigate through the model to find and display the results.
- The result display is modal, meaning you can only view one display at a time.

To address these limitations, GoldSim allows you to create ***Result elements***. A Result element is a specialized element representing a particular result display. As such, there are four types of basic Result elements: Time History, Distribution, Multi-Variate, and Array. When you double-click on a Result element, the result is directly displayed:

This is a Distribution Result element

Result2

Double-clicking on the Distribution Result element displays the result



Result elements have several important advantages:

- They allow you to access results via a simple double-click.
- They allow you to make key results readily accessible by placing them in one or more convenient locations (e.g., in a single "results" Container) to facilitate presentations and use of the model by others.
- All the result characteristics are saved with the Result element, so that you can completely control and customize how the result will be displayed when it is subsequently viewed (e.g., in a presentation or by another user). Once you display the result, the last size and position of the result display is also saved with the Result element.
- Result elements can be opened prior to the start of a simulation, and are updated dynamically as the simulation progresses (e.g., as each realization is completed).
- The display windows for Result elements are modeless, meaning that you can view multiple display windows simultaneously.

In addition, in some cases, Result elements have special capabilities that do not exist for the corresponding interactive result.

Read more: Specialized Uses for Result Elements (page 532).

Adding Result Elements to Your Model

There are two ways to create a result element: by converting an interactive result into a Result element, or by creating a Result element directly.

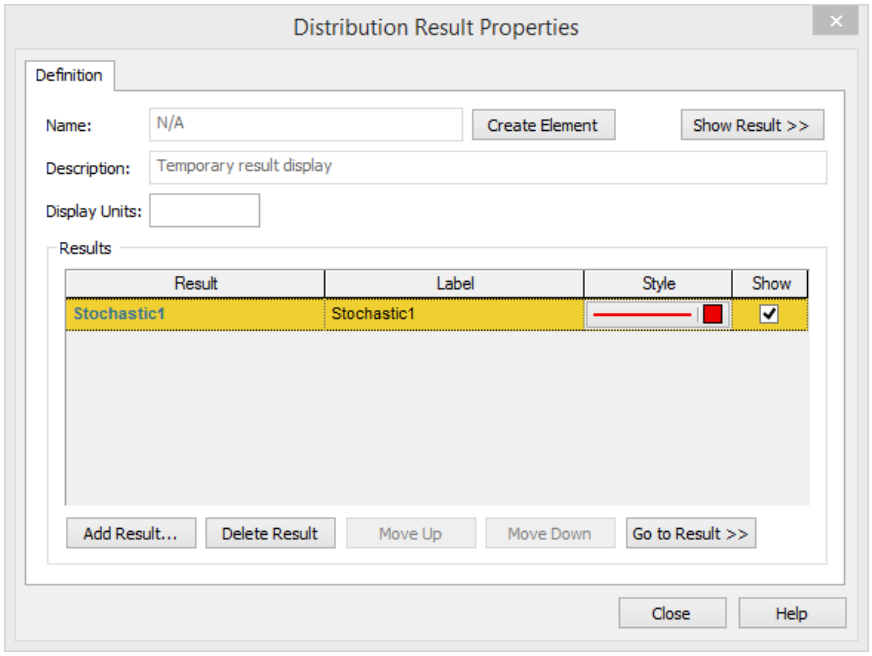
While viewing an interactive result display (in which you right-click on an element or an output and select a result display option from a context menu), you can convert it to a Result element.

***Result Properties** button*

You can do so from the Result Properties dialog for the result you are viewing.

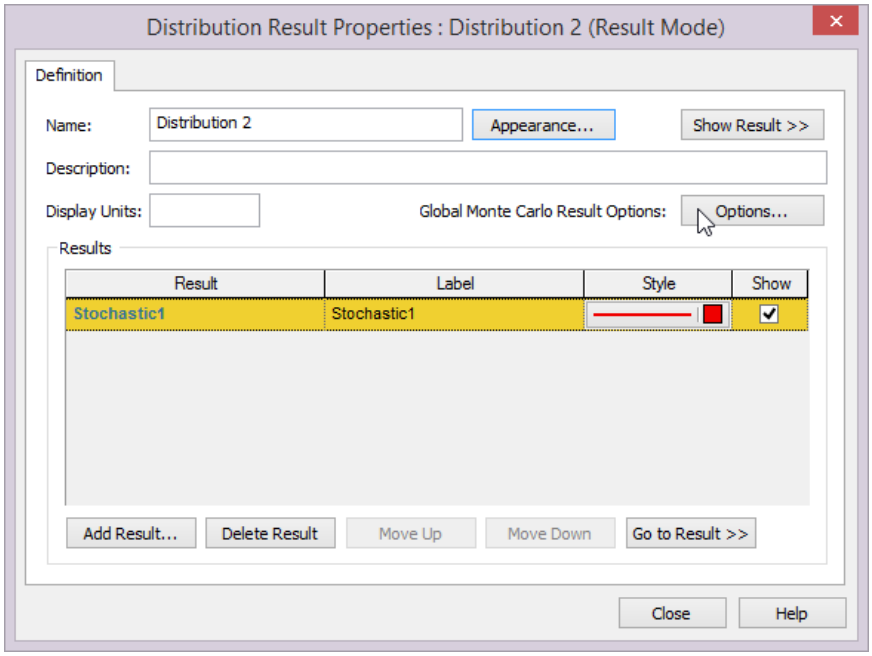
Read more: Viewing and Editing Result Properties (page 520).

This dialog can be accessed by pressing the Result properties button while viewing the chart or table. This will display the Result Properties dialog:



This is the Result Properties dialog for a Distribution result. Dialogs for other result types are similar.

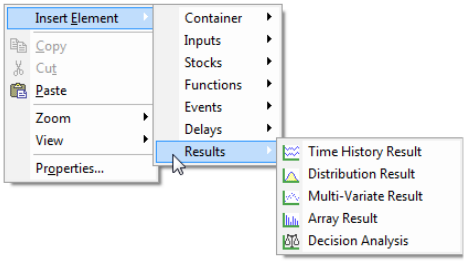
If you press the **Create Element** button, GoldSim will insert an element. The **Name** will also become editable (a default name will be automatically inserted that you can change). You will also note that the **Create Element** button changes to **Appearance**:



Note: With one exception, Result elements follow the same rules as other GoldSim elements regarding their names (e.g., can only include letters, numbers and the underscore character). Unlike other elements in GoldSim, however, the **Name** of a Result element can contain spaces.

For some result types, some additional fields may also become available when the Result element is created (in the example above, an **Options…** button becomes available).

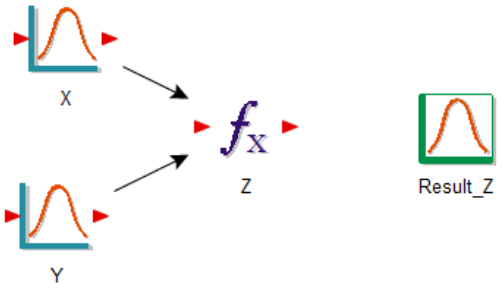
You can also insert a result element into your model directly just as you would insert any other kind of element, via the main menu or context menus:



Viewing a Result Element

Result elements are similar to other elements in GoldSim, in that you can move them, copy them and change their appearance. There are, however, several important differences:

- When you double-click on a result element in Result Mode, Run Mode or Scenario Mode, the result window is immediately displayed (double-clicking on a normal element in any Mode or a result element in Edit Mode displays its properties dialog).
- Result elements do not have input or output ports (since they do not have any outputs). Although they do reference other elements (the results being displayed), these are not considered inputs. Hence, there are not influence lines drawn to Result elements:



A result element has the same result display and editing capabilities as an interactive result display. You can choose to view a table or chart form of the result, or view the Result Properties page (e.g., in order to add other outputs to the display). Result elements "remember" the last type of result format (i.e., table or chart), and display that format the next time they are viewed (double-clicked).

Like an interactive result, you can edit the appearance of the chart (e.g., headers, footers, etc.).

Read more: Editing the Appearance of a Chart (page 661).

Note, however, that whenever you change the appearance of the chart for a result element, *these changes are automatically saved with the* element.

One of the key features of result elements is that they can be opened prior to the start of a simulation, and then viewed during the simulation. The display is updated whenever the simulation is paused, and at the end of every realization.

This can be particularly useful in helping you to debug a model (by pausing the simulation at specific points).

The dynamic display of result elements is facilitated by GoldSim's ability to slow down the speed of the simulation, step through a simulation one realization or timestep at a time, and pause and resume a simulation.

Read more: Pausing and Stepping through a Simulation (page 460).

Note that you cannot open a result element while a model is running. If you want to view a result element being dynamically updated, you must open it prior to starting the simulation, or while the simulation is paused. If you open a result element prior to starting the simulation and try to view a chart, the resizable result display window will open with a message that there is no data available.

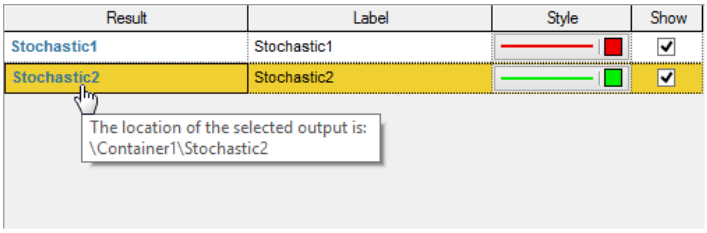
Browsing Between Result Elements and Referenced Outputs

Unlike other elements in GoldSim, when you reference an output of an element within a Result element, it is not treated as a "link". That is, the Result element is not considered to be a "function of" of the outputs that it references (since the Result element is not actually carrying out a calculation; it is simply serving as a mechanism to display a result). As a result, you will note that influences are not drawn between the element(s) being referenced and the Result element. Moreover, the Result element does not appear in the "Affects" list for the element being referenced.

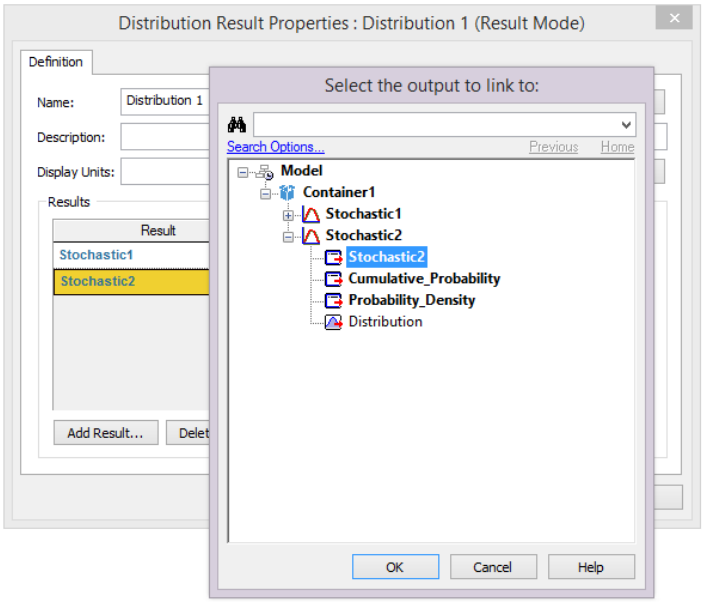
Read more: Understanding Influences (page 102); Viewing Element Dependencies (page 117).

Nevertheless, it is often necessary to quickly find the elements being referenced by a Result element (and conversely, the Result element that references a particular element). To facilitate this, GoldSim provides a number of tools for browsing between Result elements and the outputs (and hence elements) that they reference:

- From within the Properties dialog for a Result element, if you select a result (i.e., an output), and then press the **Go to Result>>** button, the dialog will close and the output's element will be selected (i.e., GoldSim will jump directly to that location in the graphics pane). (Note that Array results do not have a **Go to Result>>** button, as this button is most valuable when a Result element contains multiple results).
- From within the Properties dialog for a Result element, if you place your cursor over the result, a tool-tip will be displayed showing the location of the element being referenced:

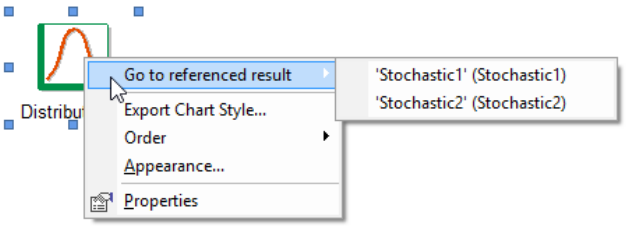


If you double-click on the result, a browser will be shown illustrating the location of the element in the hierarchy:



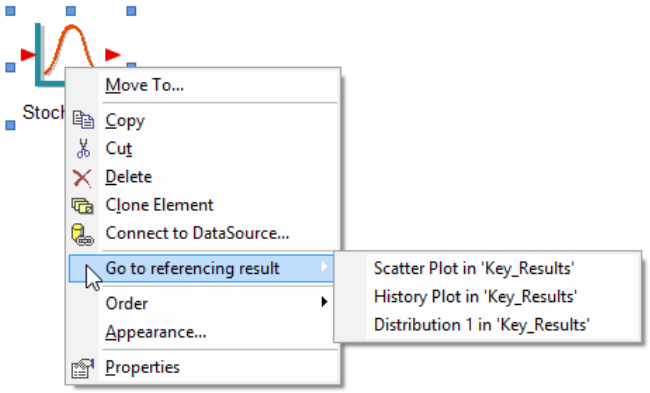
If you Ctrl+double-click on the result, the dialog will close and the output's element will be selected (i.e., GoldSim will jump directly to that location in the graphics pane).

- If your right-click on a Result element in the graphics pane (or the browser), a context menu will be displayed listing all of the referenced outputs:



If you click on one of the outputs listed, the output's element will be selected (i.e., GoldSim will jump directly to that location in the graphics pane).

- If you right-click on a referenced element in the graphics pane (or the browser), a context menu will be displayed listing all of the Result elements that reference outputs from the element:



If you click on one of the Result elements listed, the Result element will be selected (i.e., GoldSim will jump directly to that location in the graphics pane).

Specialized Uses for Result Elements

Result elements have special capabilities that do not exist for the corresponding interactive result:

- Time history results for simulations with multiple realizations can only be displayed for outputs that are referenced by Time History Result elements.

  Read more: Viewing Time Histories of Multiple Realizations (page 552); Saving Outputs as Results (page 453).

- Time History and Distribution Result elements can be used to display Scenario results.

  Read more: Viewing Scenario Results in Time History Result Elements (page 578); Viewing Scenario Results in Distribution Result Elements (page 622).

- Time History Result elements can be specified to export time history results (i.e., model outputs) to spreadsheets and text files automatically at the end of a simulation.

  Read more: Exporting from a Time History Result Element to a Spreadsheet (page 678); Exporting from a Time History Result Element to a Text File (page 686).

- Time History Result elements can be specified to plot results based on Reporting Periods.

  Read more: Viewing Reporting Period-Based Results in Time History Result Elements (page 564).

- Time History Result elements within a parent model can be used to display time history results from inside a SubModel.

  Read more: Viewing SubModel Results in Time History Result Elements (page 573).

- Distribution Result elements within a parent model can be used to display nested Monte Carlo simulation results generated using a SubModel.

  Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933).

- Time History Result elements can be specified to display results at unscheduled update points.

  Read more: Viewing Unscheduled Updates in Time History Result Elements (page 586).

Viewing Scenario Results

GoldSim's scenario modeling capability allows you to directly compare results generated by different sets of input parameters. In effect, when you use this capability, your model can store (and subsequently compare) multiple sets of results (and inputs).

Read more: Creating, Running and Comparing Scenarios (page 463).

In particular, when a model is in Scenario Mode, if you were to browse the model, you would note the following:

- There is no option to right-click on an element and view its results. Elements do not store results in Scenario Mode.

- If you double-click on a Time History Result element or a Distribution Result element, GoldSim displays results for all scenarios for which scenario results have been generated (and for which the **Show** button has been checked from within the Scenario Manager).

That is, scenario results can only be viewed in Time History and Distribution Result elements.

Read more: Viewing Scenario Results in Time History Result Elements (page 578); Viewing Scenario Results in Distribution Result Elements (page 622).

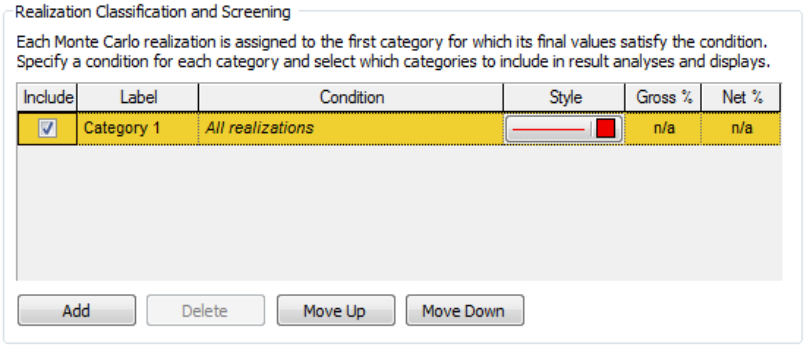
Classifying and Screening Realizations

When carrying out probabilistic simulations, you may often run hundreds or thousands of realizations. In order to analyze the results, it is often quite useful to classify the realizations into *categories*. A category is simply defined by a condition relating one or more outputs in the model (e.g., those realizations in which the discount rate was above 3.5%; those realizations in which the profit exceeded $1,000,000; those realizations in which the peak concentration was between 1 mg/l and 10 mg/l).

The power of categories is that once you have defined them, you can use them in two ways:

- Within certain kinds of result charts (e.g., scatter plots, time histories, distributions), realizations from each category can be displayed in a different color (and/or symbol).
- Within all result displays, you can choose to screen out one or more categories, so that the results that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include.

Classification categories are defined at the bottom of the Monte Carlo Result Display Properties dialog:

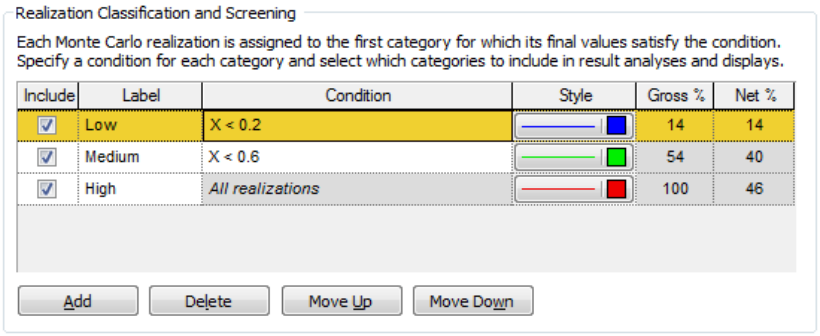


This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the **Result Options…** button), and is also accessible from result properties dialogs.

Read more: Viewing and Editing Result Properties (page 520).

By default, all results are placed in a single category (defined as "All realizations"). You can add categories by pressing the **Add** button. When you do so, it adds a row above the selected row. For each category that you add, you must define a **Label** and a **Condition**.

In this example, three categories have been defined:



It is critical to understand how the categories of realizations are created by GoldSim. ***In particular, the order of the Classification Conditions is important.*** If multiple Classification Conditions are true for a realization, the realization is assigned to the category with the first True Condition in the list. Hence, in the example shown above:

- The "Low" category consists of all realizations in which X < 0.2. As indicated by the **Net %** column, 14% of the realizations fall into this category.
- The "Medium" category consists of all realizations in which X >= 0.2 (and hence they do not fall into the "Low" category above it) AND X < 0.6. As indicated by the **Net %** column, 40% of the realizations fall into this category.
- The "High" category consists of all remaining realizations (all realizations in which the conditions in the first two categories are false). In this case, it implies that the category consists of all realizations in which X >= 0.6. As indicated by the **Net %** column, 46% of the realizations fall into this category.

As pointed out above, the **Net %** columns indicates the percentage of realizations falling into each categroy, The **Gross %** column shows the percentage of realizations falling into the selected category and all the categories above it.

Note: If you place your cursor over the **Net %** field, a tool-tip will display the actual realizations that fall into that category; if you place your cursor over the **Gross %** field, a tool-tip will display the actual realizations that fall into that category and all the categories above it.

Note: The **Gross %** and **Net %** are only shown when the model is in Result Mode (or when the model is paused).

Read more: Understanding Result Mode (page 512).

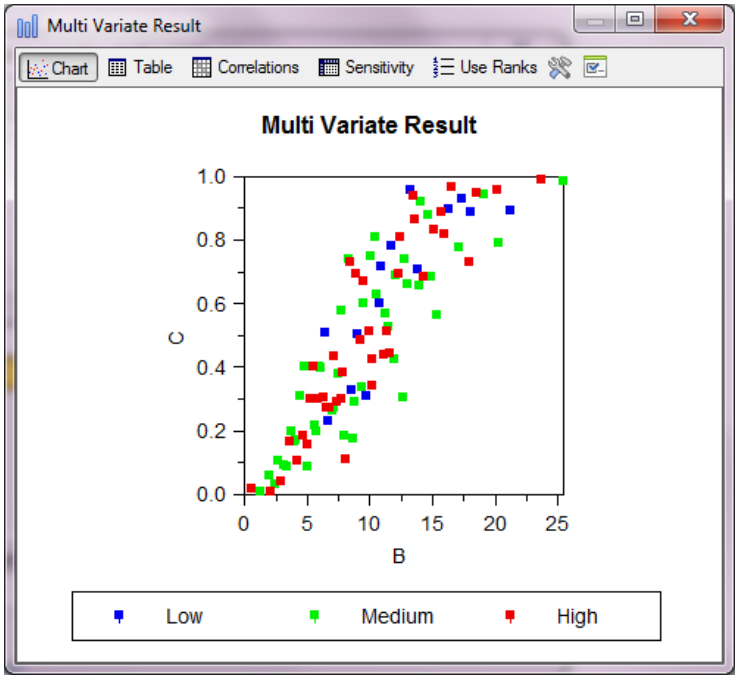
You can use the **Move Up** and **Move Down** buttons to change the order of the Conditions (and hence the definition of the categories).

Several points should be noted regarding defining the categories:

- You cannot edit the Condition for (or move) the final category. It always represents "everything else". That is, it represents all realizations that do not fall into one of the previous categories.

- The Condition can be a complex expression, but it must be a scalar condition.
- You can edit the Conditions in Edit Mode, Result Mode or Run Mode (but not Scenario Mode). However, in Result Mode or Run Mode, you cannot reference an output for which Final Values have not been saved.

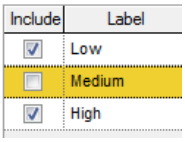
Categories are powerful because in Result Mode, within certain kinds of result charts (e.g., scatter plots, time histories, distributions), realizations from each category can be displayed in a different color (and/or symbol). For example, this Multi-Variate Result (a 2D scatter plot) breaks down the results by category:



Note that for each category, you can select a **Style**. This is used when plotting the results when you have created multiple categories. The manner in which these Styles are used is a function of the type of result being plotted.

Read more: Using Result Classification and Screening in Time History Results (page 571); Using Result Classification and Screening in Distribution Results (page 616); Using Result Classification and Screening in Multi-Variate Results (page 643); Using Result Screening in Array Results (page 658).

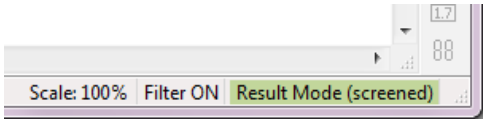
In addition, you can choose to screen out one or more categories of realizations by clearing the **Include** checkbox for a category:



When you do so, when showing results (in both charts and tables), only those realizations in the categories which you have chosen to include are displayed.

Note: You can edit the **Include** box in Edit Mode, Result Mode Scenario Mode or Run Mode.

Obviously, if you have screened out results in such a way, it is critical that this is made apparent that the results being shown are screened. Hence, when results are screened, this is indicated in the right-hand corner of the status bar:



Creating Chart Styles

GoldSim has powerful charting capabilities that allow you to customize the appearance of each chart (e.g., by adding headers and footers, changing axes scales and labels, etc.).

Read more: Editing the Appearance of a Chart (page 661).

In some cases, after modifying the appearance of a chart, you may want to save the particular set of properties you have created (e.g., label fonts, header, footer) so that you can apply them to another chart. To facilitate this, GoldSim allows you to save the set of properties as a named ***chart style***. The named chart style is saved as part of the model file, and can subsequently be applied to other results in your model. You can also export and import chart styles between models.

In order to facilitate the use of a single chart style for multiple results, GoldSim allows you to use ***keywords*** (delimited by the % symbol) when you create and use chart styles. For example, you could define a chart style in which the header was %rname%. This particular keyword inserts the name of the Result element you are displaying. Similarly, the keyword %curdate% inserts the current date.

When you view a result, it will initially take on a set of characteristics defined by the default chart style for that particular type of chart (e.g., time history, distribution, etc.). GoldSim automatically provides a set of default chart styles for all chart types (that utilize the keywords). If desired, however, for any type of chart, you can specify one of your own custom chart styles as the default chart.

Read more: Creating and Using Chart Styles (page 671).

Viewing Time History Results

Time History results show the "history" of a particular output as a function of time, and are probably the most common form of result display you will use.

A Time History result has two types of views:

- Chart View; and
- Table View.

If you have saved Time Histories for an output, you can display a Time History result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Time History Result…** from the context menu. By default, a chart will be displayed.

The specific time points that are displayed in Time History results are determined by your simulation settings.

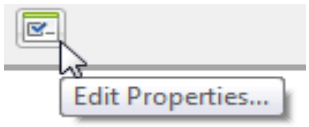
Read more: Specifying When Time History Results Will Be Saved (page 423).

Note: Time history results for simulations with multiple realizations can only be displayed for outputs that are referenced by Time History Result elements.

Viewing the Properties of a Time History Result

The topics below discuss the display and use of Time History results in detail.

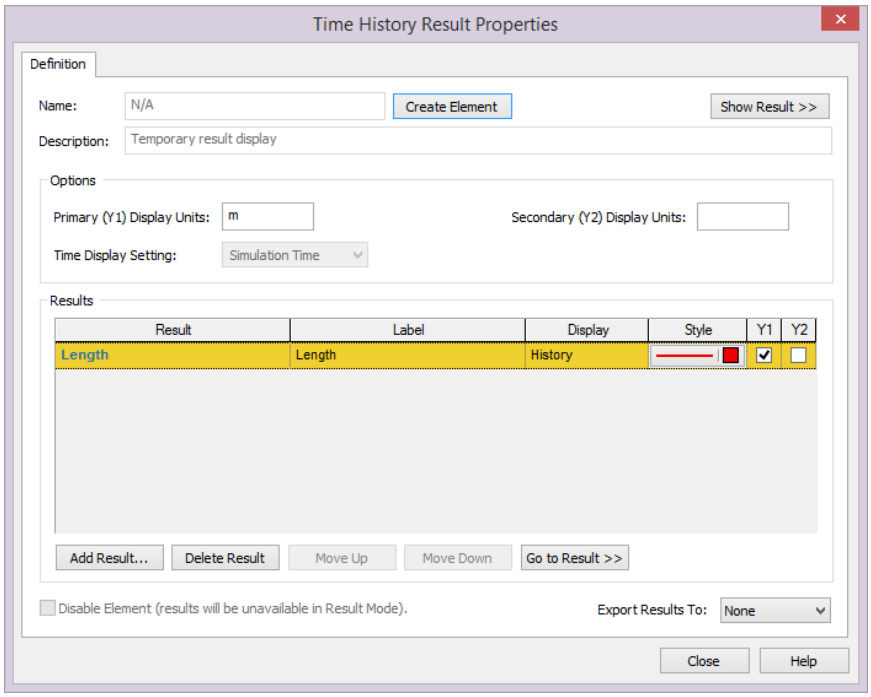
If you click on the Result Properties button when viewing a Time History Result chart or table, the Properties dialog for the result will be displayed. The Result Properties button is the furthest button to the right at the top of the display window:



Note that when you press this button, it does not close the result display window you were displaying. Both windows are displayed simultaneously (although you can subsequently close one or the other). When viewing results, any changes you make to the Result Properties are immediately represented in the result display.

The Result Properties dialog is always modal. That is, it cannot be minimized, and with the exception of viewing the result display (which can be displayed simultaneously and share the focus with the Properties dialog), it retains the focus while it is displayed. As a result, you must close the Result Properties dialog before you can edit any other part of your GoldSim model.

The Result Properties dialog for an interactive (scalar) Time History result looks like this:



You can show the full path for a result (showing the containment hierarchy) by placing your cursor over the Result item to display a tool-tip.

At the top of the page is a button to **Create Element**. By pressing this button, a Result element is created. When you do so, the key change that will be noticeable is that the **Name** will now be editable. The **Create Element** button becomes an **Appearance** button (which is used to modify the appearance of the element itself).

Read more: Creating and Using Result Elements (page 526).

The **Show Result >>** button opens a result display window (while keeping the Properties dialog simultaneously open.

Read more: Viewing a Time History Chart (page 540); Viewing a Time History Table (page 542).

The **Add Result…** button allows you to add results to the chart. Results can be deleted with the **Delete Result** button. The order in which multiple results appear in the list can be changed using the **Move Up** and **Move Down** buttons.

Read more: Viewing Time Histories of Multiple Outputs (page 544).

For each result in the list, you specify a **Style** (used in charts), and the **Label** used in legends in charts and headers in tables.

Read more: Controlling the Chart Style in Time History Results (page 592).

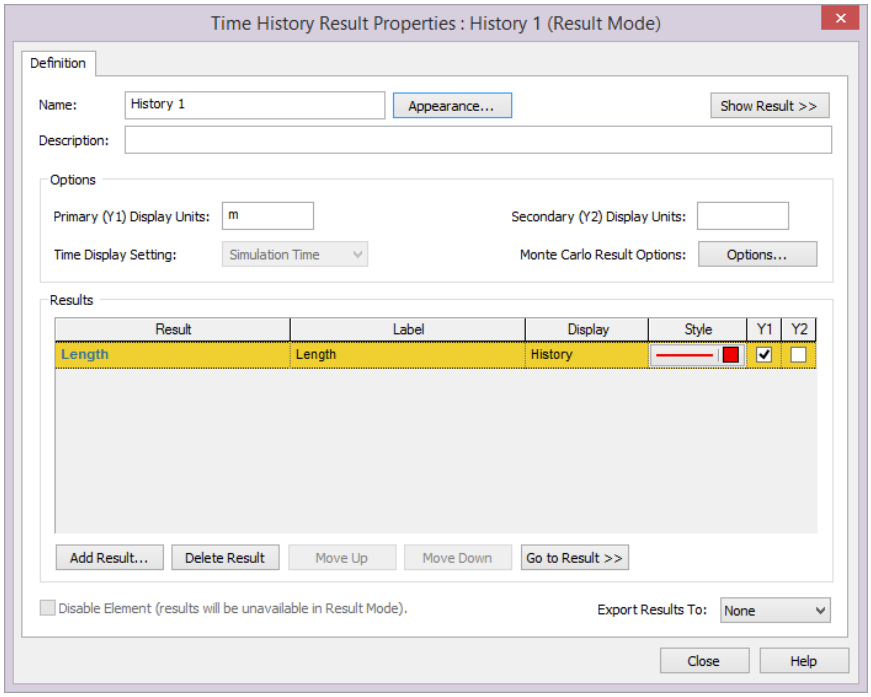
Note: The colors used for a Probability chart (which shows the percentiles for a multi-realization simulation) are not specified by the **Style** button. Rather, these are controlled via the Global Monte Carlo Result Options.

Read more: Viewing Probability Histories for Multiple Realizations (page 555).

The **Y1** and **Y2** checkboxes are used to determine which axis the result is plotted on. If you add a second output that has the same dimensions as the original output, it will automatically be placed on the primary (Y1) axis. If it has different dimensions, it will automatically be assigned to the secondary (Y2) axis. You can add as many outputs as desired to the list. If you add an output whose dimensions do not match those currently assigned to the Y1 or Y2 axes, it will be assigned to neither. As a result, it will not be displayed in the chart (but will be displayed in the table). If you specifically try to assign an unassigned output to an axis, any outputs assigned to that axis with different dimensions will be cleared and the axis will take on the new dimension.

You can also specify the **Display Units** for the Y1 and Y2 axis (which overrides the display units specified within the element's property dialog).

The Properties dialog for a Time History Result element has several differences:



- The **Create Element** button is replaced by an **Appearance** button (which is used to modify the appearance of the element itself).
- The **Name** (and **Description**) are editable (since this is an element).
- An **Options…** button is available that provides access to the Monte Carlo Result Display Properties.

  Read more: Controlling Monte Carlo Result Options (page 441).
- Time History Result elements also have two additional options at the bottom of the dialog: the ability to disable the element, and the ability to export results (to a spreadsheet or text file) at the end of a simulation.

  Read more: Disabling a Time History Result Element (page 591); Exporting From a Time History Result Element to a Spreadsheet (page 678); Exporting from a Time History Result Element to a Text File (page 686).
- The **Time Display Setting** can only be edited while in Edit Mode. It defaults to "Simulation Time", which is the appropriate setting for viewing standard time histories. However, two additional options are also available ("Reporting Periods" and "SubModel Time") which can be selected in order to view Reporting Period-based results and SubModel results, respectively, in Time History Result elements.

  Read more: Viewing Reporting Period-Based Results in Time History Result Elements (page 564); Viewing SubModel Results in Time History Result Elements (page 573).

If you have run multiple realizations, the Properties dialog for a Time History Result dialog has one additional field that is not available for single realization runs: you can specify a **Custom Statistic** for each output:

This can then be viewed in result displays.

Read more: Viewing Custom Statistics for Multiple Realizations (page 560).

Viewing a Time History Chart

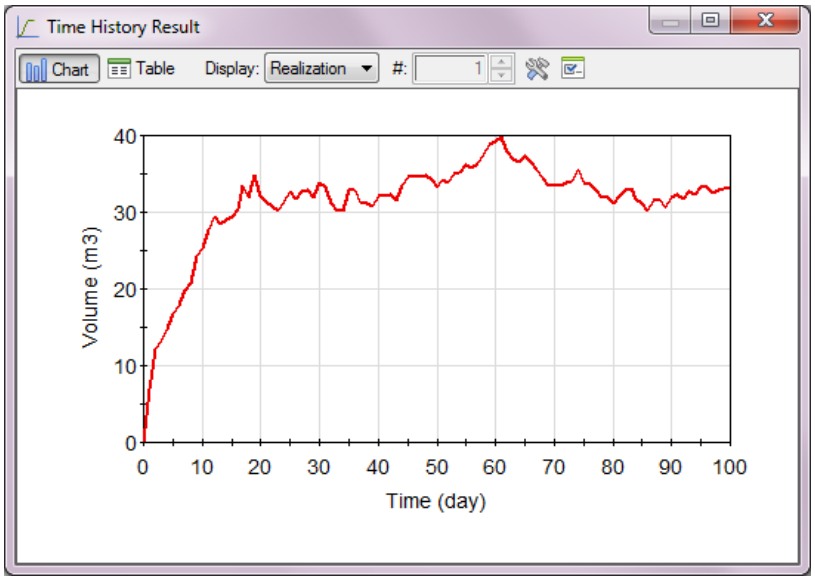
If you have saved Time Histories for an output, you can display a Time History chart by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Time History Result…** from the context menu. By default, a chart will be displayed.

Note: When viewing a Time History Result element, the element "remembers" the last type of view that was displayed, and displays that view when you double-click on it.

Note: Time history results for simulations with multiple realizations can only be displayed for outputs that are referenced by Time History Result elements.

If you are viewing a table rather than a chart, you can view a chart by pressing the **Chart** button at the top of the display.

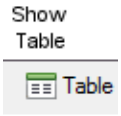
A Time History Chart looks like this:



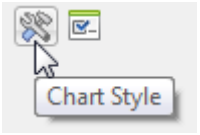
The Time History Chart has a variety of buttons and controls at the top of the window. Some items are always present, while others are only shown under certain circumstances (e.g., when viewing multiple realizations and/or multiple outputs).

The following buttons are always available:

Show Table: Selecting this button switches to a Time History Table view of the result. Note that when viewing a table, the Show Table button appears selected; when viewing a chart, the Show Chart button appears selected.

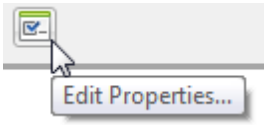


Edit Chart Style: This button provides access to a dialog for editing the chart style.



Read more: Controlling the Chart Style in Time History Results (page 592).

Edit Properties: This provides access to the Result Properties.



Read more: Viewing the Properties of a Time History Result (page 537).

In some situations, you may want to zoom in on a portion of a chart that is of particular interest. You can, of course, edit the style of the chart and change the range of the axes which are displayed. GoldSim also provides a keyboard short-cut to support this.

Read more: Zooming in on a Chart (page 523).

Condition outputs are either True or False and are typically used as state variables or flags in a simulation. As a result, in some situations, you may want to save and plot time histories of conditions. To facilitate this, when plotting a time history of a condition, GoldSim plots True as 1 and False as 0.

Like all result charts, you can also control various attributes of the chart via a context menu. This includes the ability to turn on and off a legend. The legend uses the **Label** for the result defined in the Result Properties page. When viewing multiple results on one chart, you will always want to display the legend.

Read more: Using Context Menus in Charts (page 522); Viewing Time Histories of Multiple Outputs (page 544).

You can control the number of significant figures displayed in result displays from the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).

Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 524).

You can copy the chart to the clipboard by right-clicking in the chart, and selecting **Copy** from the context menu. Alternatively, you can press **Ctrl+C** while viewing the chart. To export a chart, right-click in the chart, and select **Export…** from the context menu.

Read more: Copying a Chart or Table (page 691); Exporting a Chart (page 691).

Note: When viewing time history results from a SubModel within the parent model, the Chart display is modified somewhat, and in some cases provides slightly different options.

Read more: Viewing SubModel Results in Time History Result Elements (page 573); Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933).

Viewing a Time History Table

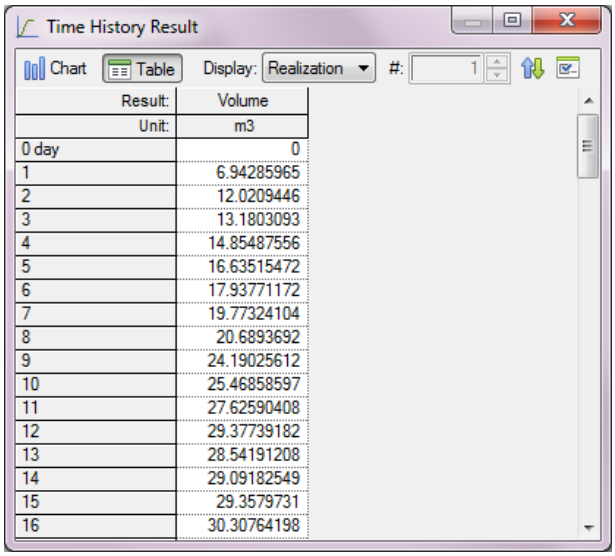
When you display a Time History result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Time History Result…** from the context menu., by default, a chart will be displayed.

Read more: Viewing a Time History Chart (page 540).

You can view a Time History Table by pressing the **Show Table** button when viewing a chart.

Note: When viewing a Time History Result element, the element "remembers" the last type of view that was displayed, and displays that view when you double-click on it.

A Time History Table looks like this:



Result:	Volume
Unit:	m3
0 day	0
1	6.94285965
2	12.0209446
3	13.1803093
4	14.85487556
5	16.63515472
6	17.93771172
7	19.77324104
8	20.6893692
9	24.19025612
10	25.46858597
11	27.62590408
12	29.37739182
13	28.54191208
14	29.09182549
15	29.3579731
16	30.30764198

For multi-reallization runs, there are three header rows. The first shows the Label for the result; the second shows the unit; and the third identifies exactly what is being displayed (e.g., a realization or a specific statistic). For single realization runs, the third header row is not shown (as it is not applicable).

The first column then shows the plot points being used (i.e., scheduled timesteps at which point values are saved). Each row represents a separate plot point (a result value that was saved at a particular point).

What is displayed in the first column is a function of two things:

- Is the **Time Display Setting** (in the Properties dialog) set to "Simulation Time", "SubModel Time" or "Reporting Periods"? In the first two cases ("Simulation Time is the default and the only option for interactive results), the first column contains elapsed times or dates. In the last case, the first column contains Reporting Period names.

  Read more: Viewing Reporting Period-Based Results in Time History Result Elements (page 564).

- Is the **Time Basis** in the Simulation Settings dialog "Elapsed Time" or "Calendar Time"? In the former case, the first column displays elapsed times. In the latter case, the first column displays dates.

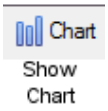
  Read more: Defining the Time Basis and Simulation Duration (page 414).

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively. If the column is too narrow to display the contents of a cell, holding the cursor over the cell displays a tool-tip with the contents.

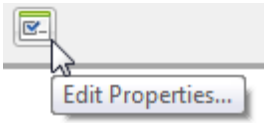
The Time History Table has a variety of buttons and controls at the top of the window. Some of items are always present, while others are only shown under certain circumstances (e.g., when viewing multiple realizations and/or multiple outputs).

The following three buttons are always available:

Show Chart: Selecting this button switches to a Time History Chart view of the result. Note that when viewing a table, the Show Table button appears selected; when viewing a chart, the Show Chart button appears selected.

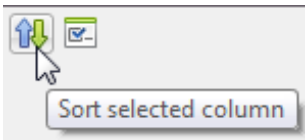


Edit Properties: This provides access to the Result Properties.



Read more: Viewing the Properties of a Time History Result (page 537).

Sort: This button allows you to sort the rows based on a selected column.



Read more: Sorting Values in Result Tables (page 524).

Several additional points should be noted regarding Time History Tables:

- Conditions are displayed in tables as either 1/0, true/false, True/False, TRUE/FALSE, on/off, or High/Low. You select which of these pairs to use on the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).

- You can control the number of significant figures displayed in tables from the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).

  Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 524).

- You can copy the contents of a Time History Table to the clipboard by pressing **Ctrl+C**. You must first select the cells you wish to copy. You can do so by placing your cursor in one cell and dragging to another location. You can select the entire table by pressing the cell in the upper left-hand corner of the table. Cells that are not adjacent can be selected by pressing the **Ctrl** key when selecting. Selected items will be highlighted in black. You can subsequently paste the table into another application (such as a spreadsheet).

  Read more: Selecting Items and Copying Values in Result Tables (page 523).

Note: When viewing time history results from a SubModel within the parent model, the Table display is modified somewhat, and in some cases provides slightly different options.

Read more: Viewing SubModel Results in Time History Result Elements (page 573); Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933).

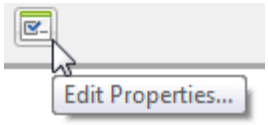
Viewing Time Histories of Multiple Outputs

It is often useful to plot the time histories of multiple outputs on a single plot. This allows you to compare and contrast different results, or to better understand how various parameters in your model affect each other. Therefore, GoldSim allows you to display multiple outputs on a single time history chart or table.

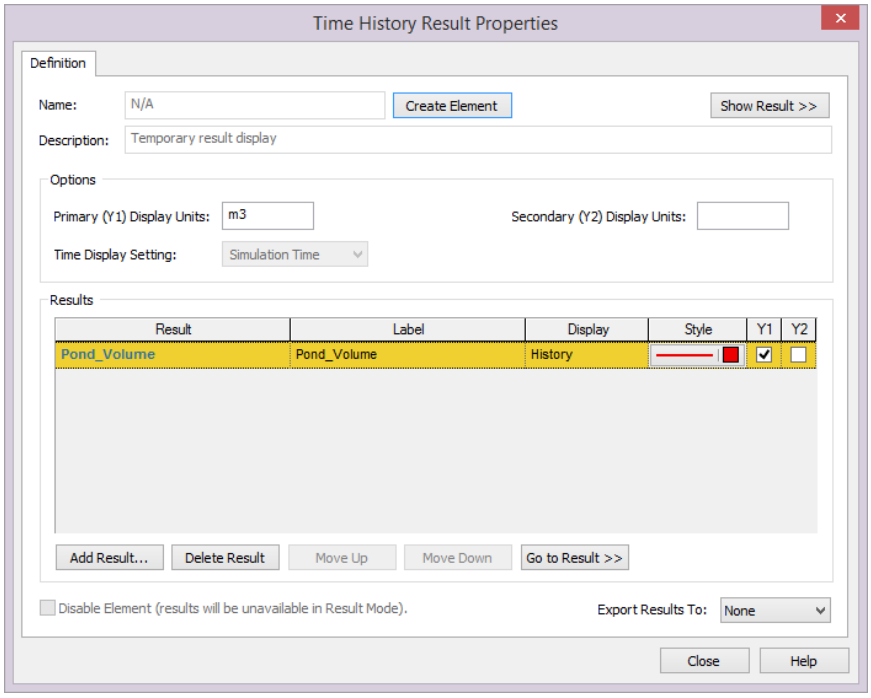
To display multiple results in a Time History Result, you must first open the Result Properties dialog for the result.

Read more: Viewing the Properties of a Time History Result (page 537).

You can do this by pressing the **Edit Properties…** button in any of the Result display windows:

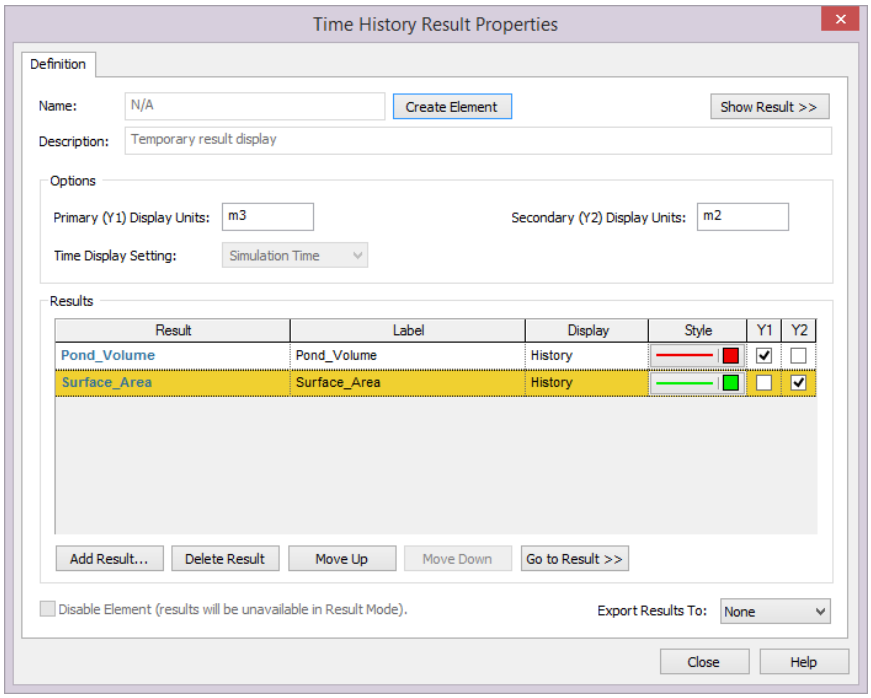


The following dialog will be displayed:



This is the dialog for an interactive result. A Result element will have a Name defined. Note that the Label is user-editable and is used in legends and column headers.

To add additional outputs to the result, press the **Add Result…** button. A dialog for selecting an output will be displayed. After you select the output you wish to add to the result display, and press **OK**, the selected outputs are then shown in the Result Properties dialog:



You can show the full path for each result (showing the containment hierarchy) by placing your cursor over the Result item to display a tool-tip.

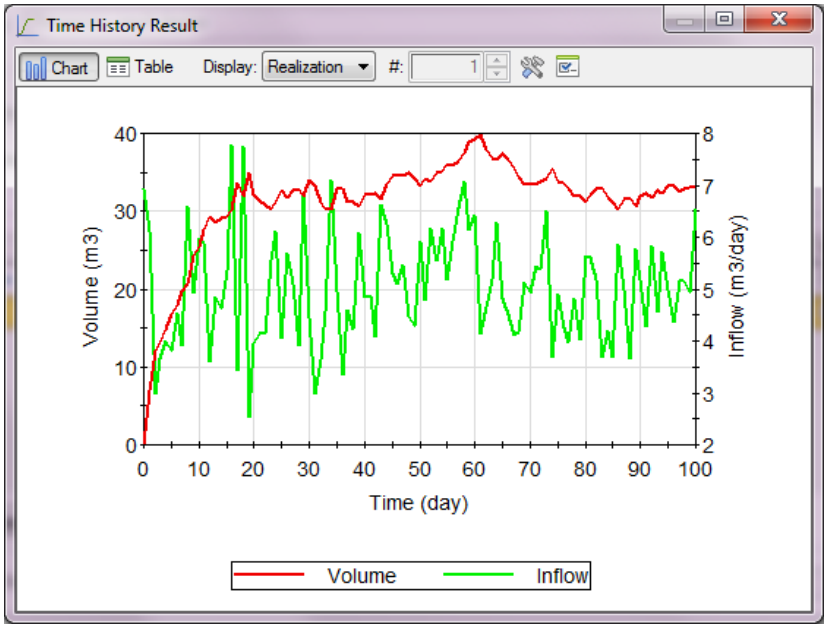
Results can be deleted with the **Delete Result** button (holding the Ctrl key down changes this to a **Delete All** button). The order in which multiple results appear in the list can be changed using the **Move Up** and **Move Down** buttons.

The **Show Result >>** button opens a result display window (while keeping the Properties dialog simultaneously open).

Read more: Viewing a Time History Chart (page 540); Viewing a Time History Table (page 542).

Pressing the **Show Result** button displays the multiple output display.

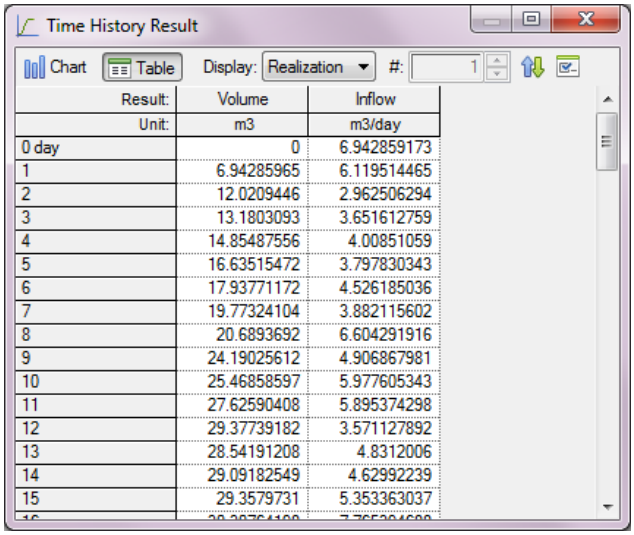
A Time History Chart with multiple outputs looks like this:



The legend displays the (user-editable) Labels defined for each result.

Note that a legend is available for Time History Charts (you may have to turn it on by right-clicking in the chart and selecting to show the legend from the context menu). The Labels specified in the Result Properties dialog are used in the legend to label the different results.

For Time History Tables, the results are listed in separate columns:



Result:	Volume	Inflow
Unit:	m3	m3/day
0 day	0	6.942859173
1	6.94285965	6.119514465
2	12.0209446	2.962506294
3	13.1803093	3.651612759
4	14.85487556	4.00851059
5	16.63515472	3.797830343
6	17.93771172	4.526185036
7	19.77324104	3.882115602
8	20.6893692	6.604291916
9	24.19025612	4.906867981
10	25.46858597	5.977605343
11	27.62590408	5.895374298
12	29.37739182	3.571127892
13	28.54191208	4.8312006
14	29.09182549	4.62992239
15	29.3579731	5.353363037

The column headers are the (user-editable) Labels defined for each result.

The following points should be noted regarding the Time History Result Properties dialog when displaying multiple results:

- For each result in the list, you specify a **Style** (used in charts), and the **Label** used in legends in charts and headers in tables.

  Read more: Controlling the Chart Style in Time History Results (page 592).

- The **Y1** and **Y2** checkboxes are used to determine which axis the result is plotted on. If you add a second output that has the same dimensions as the original output, it will automatically be placed on the primary (Y1) axis. If it has different dimensions, it will automatically be assigned to the secondary (Y2) axis. You can add as many outputs as desired to the list. If you add an output whose dimensions do not match those currently assigned to the Y1 or Y2 axes, it will be assigned to neither. As a result, it will not be displayed in the chart (but will be displayed in the table). If you specifically try to assign an unassigned output to an axis, any outputs assigned to that axis with different dimensions will be cleared and the axis will take on the new dimension.

- You can specify the **Display Units** for the Y1 and Y2 axis (which overrides the display units specified within the element's property dialog).

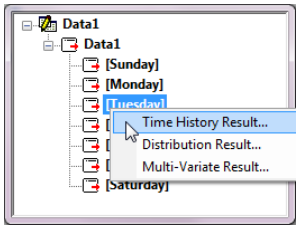
- When you are viewing time histories of multiple realizations of multiple outputs, the display windows provide a number of options that allow you to select how you want to view the results. Some options allow you to view probabilistic results for one result at a time (e.g., all realizations for a single result), while others allow you to view probabilistic results for all results simultaneously (e.g., a particular statistic for each result).

  Read more: Viewing Time Histories of Multiple Realizations for Multiple Outputs (page 561).

Viewing Time Histories for Array Outputs

When you wish to view time history results for an array output, you must first choose whether you wish to view results for a single item of the array, for selected items, or for all items.

If you wish to view results for only a single item, you should expand the array in the browser or output interface, right-click on the specific item which you wish to view, and select **Time History Result…** from the context menu.

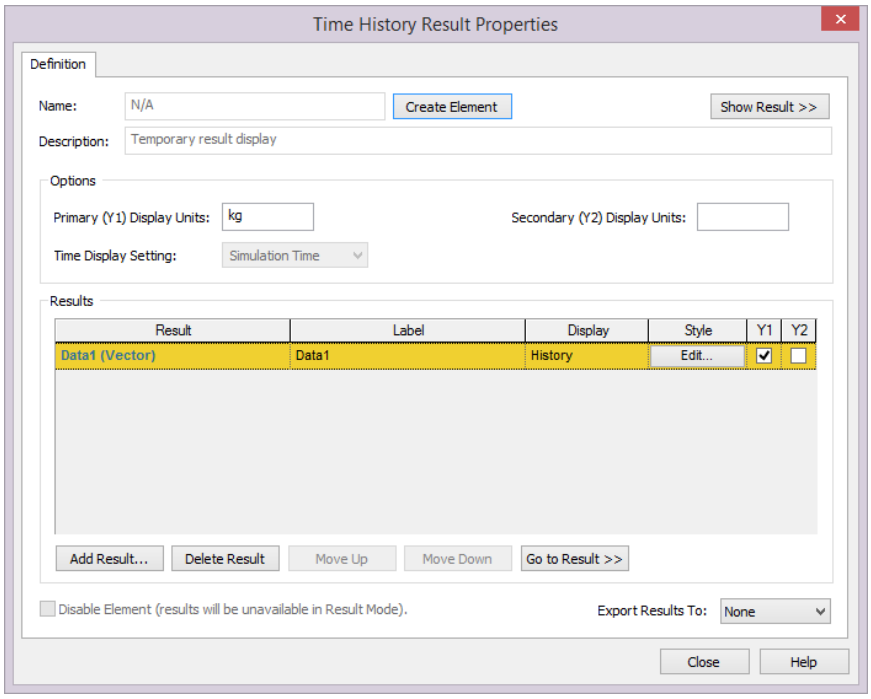


Alternatively, you could do the same thing in the Result Properties dialog (i.e., select a single item) when adding the output to the list.

Note: When you create a new array element, the **Time History** checkbox defaults off (and hence time histories are not saved). If you wish to save time histories for an array, you must specifically check this box (or reference the output in a Time History Result element).

Read more: Specifying Results to be Saved (page 453).

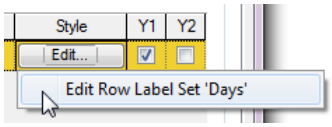
If, however, you wish to view selected items (or all items) of the array, you should either right-click on the entire array, and select **Time History Result…** from the context menu, or add the entire array to the list in the Result Properties dialog. If you do so, the Result Properties dialog looks like this:



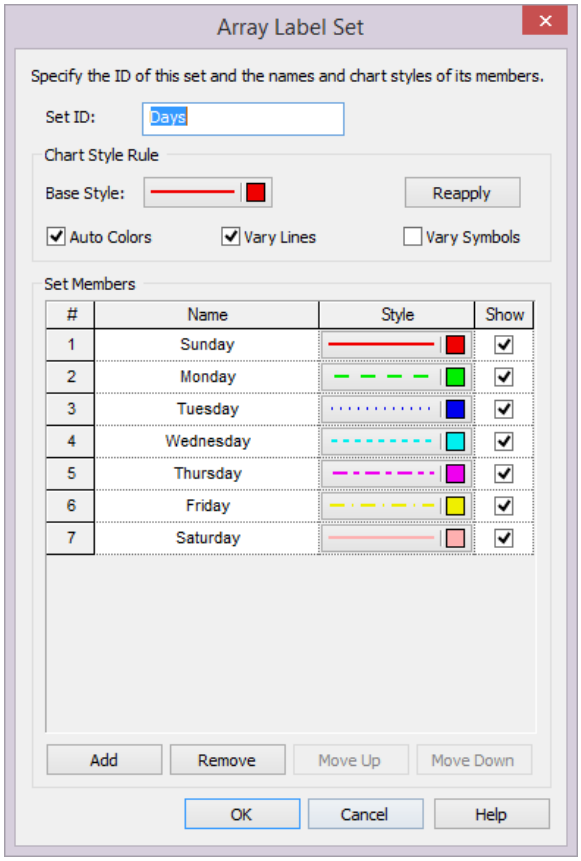
Note that in the **Result** column, the type of array (in this case, a vector) is specified.

In addition, the **Style** column includes an **Edit…** button for each array result. This button can be used to specify the Style for each item being plotted (e.g., line color), and can also be used to specify which items will be included in the display.

Pressing the **Edit…** button for a vector result displays an option to edit the Array Label set defining the vector:



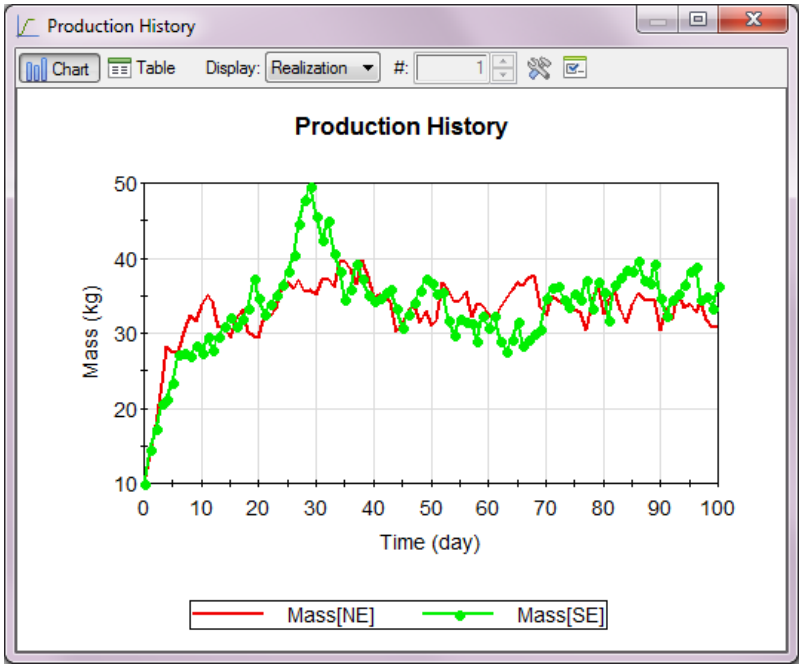
In this example, selecting this option would display the "Days" Array Label Set:



Read more: Creating and Editing Array Labels (page 728); Controlling the Chart Style in Time History Results (page 592).

This dialog allows you to change the **Style** for each item used for charts. In addition, the **Show** checkbox allows you to select which items are included in the displays (by default all will be checked).

If you choose to display a time history chart of an entire vector, the chart will look something like this.

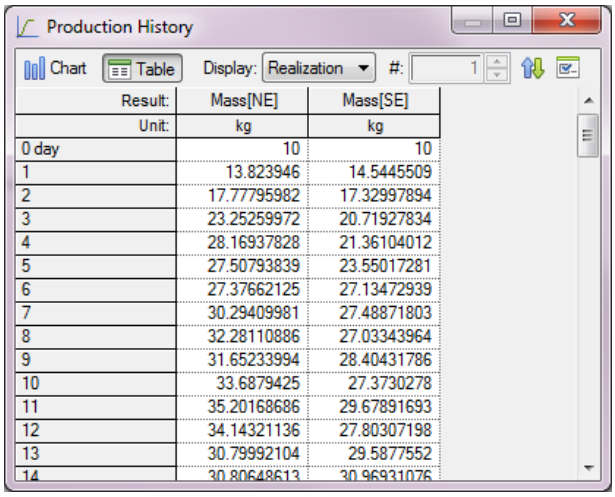


In this example, the vector has two items (named NE and SE).

GoldSim will display each item of the vector, and add a legend. The style of each line is defined by the **Style** in the Array Label Set dialog, and the legend lists the array item names.

Note: If you do not want to repeat the name of the array (Mass in the example shown above), you can specify a blank Label for that result item in the Properties dialog.

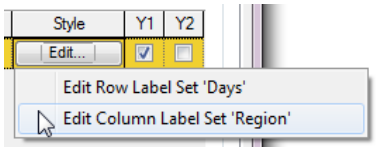
The table view of a vector would look like this:



Result:	Mass[NE]	Mass[SE]
Unit:	kg	kg
0 day	10	10
1	13.823946	14.5445509
2	17.77795982	17.32997894
3	23.25259972	20.71927834
4	28.16937828	21.36104012
5	27.50793839	23.55017281
6	27.37662125	27.13472939
7	30.29409981	27.48871803
8	32.28110886	27.03343964
9	31.65233994	28.40431786
10	33.6879425	27.3730278
11	35.20168686	29.67891693
12	34.14321136	27.80307198
13	30.79992104	29.5877552
14	30.80648613	30.96931076

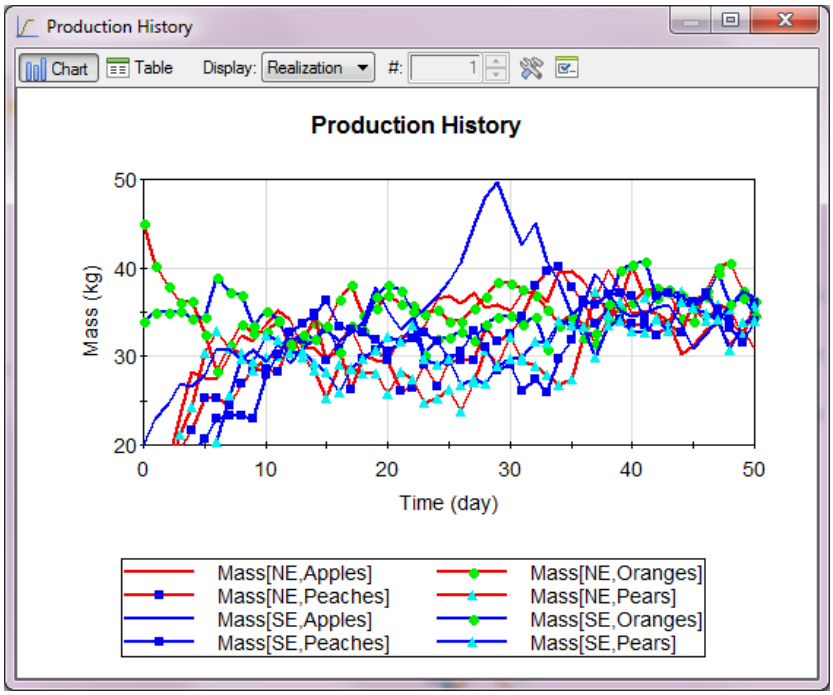
Each item of the vector is displayed in a different column.

Pressing the **Edit…** button for a matrix result displays an option to edit both the row and the column Array Label sets defining the matrix:



This allows you to select specific rows or columns to include/exclude in the displays, and also allows you to define the **Style** for each item in charts.

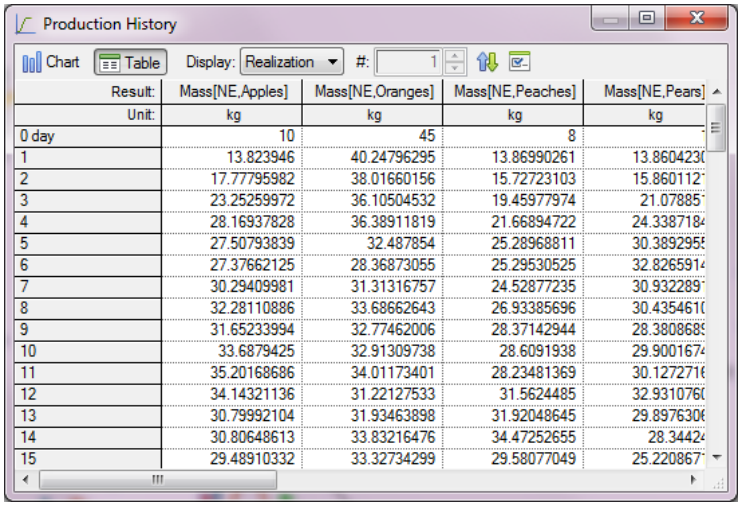
When displaying charts and tables for a matrix, GoldSim, by default, displays all items. For example, the chart below displays a matrix with 2 rows and 4 columns:



In such a chart, how are the Styles determined, since each row and each column has a specified **Style** (defined in the Array Label Set dialog) and each item being plotted is associated with both a row and a column? When plotting a matrix, the following rules are applied for each item:

- The color for an item is based on the **Style** for the row; and
- All other attributes, such as the line width, style (e.g., solid, dashed), symbol style, symbol size and symbol color, are based on the **Style** for the column.

The table view of a matrix looks like this:



Result:	Mass[NE,Apples]	Mass[NE,Oranges]	Mass[NE,Peaches]	Mass[NE,Pears]
Unit:	kg	kg	kg	kg
0 day	10	45	8	
1	13.823946	40.24796295	13.86990261	13.860423
2	17.77795982	38.01660156	15.72723103	15.860112
3	23.25259972	36.10504532	19.45977974	21.07885
4	28.16937828	36.38911819	21.66894722	24.338718
5	27.50793839	32.487854	25.28968811	30.389295
6	27.37662125	28.36873055	25.29530525	32.826591
7	30.29409981	31.31316757	24.52877235	30.932289
8	32.28110886	33.68662643	26.93385696	30.435461
9	31.65233994	32.77462006	28.37142944	28.380868
10	33.6879425	32.91309738	28.6091938	29.900167
11	35.20168686	34.01173401	28.23481369	30.127271
12	34.14321136	31.22127533	31.5624485	32.931076
13	30.79992104	31.93463898	31.92048645	29.897630
14	30.80648613	33.83216476	34.47252655	28.3442
15	29.48910332	33.32734299	29.58077049	25.220867

Each item of the matrix (which has been selected to be shown in the Array Label Set dialogs) is displayed in a different column of the table.

When you are viewing time histories of multiple realizations of an array, the display windows provide a number of options that allow you to select how you want to view the results. Some options allow you to view probabilistic results for one item at a time (e.g., all realizations for a single result), while others allow you to view probabilistic results for all items simultaneously (e.g., a particular statistic for each result).

Read more: Viewing Time Histories of Multiple Realizations for an Array (page 562).

Viewing Time Histories of Multiple Realizations

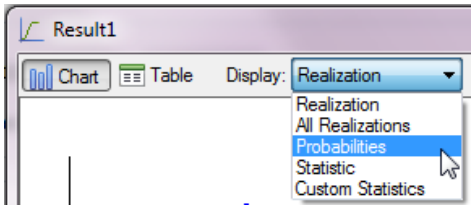
When you are viewing time histories after you have run multiple realizations, there are a number of different ways that you can view the results (e.g., view one realization at a time, view all realizations, view a particular statistic). The Display windows provide a number of options that allow you to select how you want to view the results.

Note: Time History results for simulations with multiple realizations can only be displayed for outputs that are referenced by Time History Result elements.

Note: You can only view those realizations which are available (i.e., those which have been saved and have not been screened out).

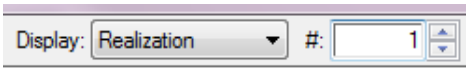
Read more: Specifying Results to be Saved (page 453); Using Result Classification and Screening in Time History Results (page 571).

If you have run multiple realizations, the top portion of the display window (either a chart or a table) for a Time History result looks like this:



As can be seen, the **Display** drop-list provides five options:

- **Realization**: This displays a single realization at a time. A control is added to the display window to select any (unscreened) realization:



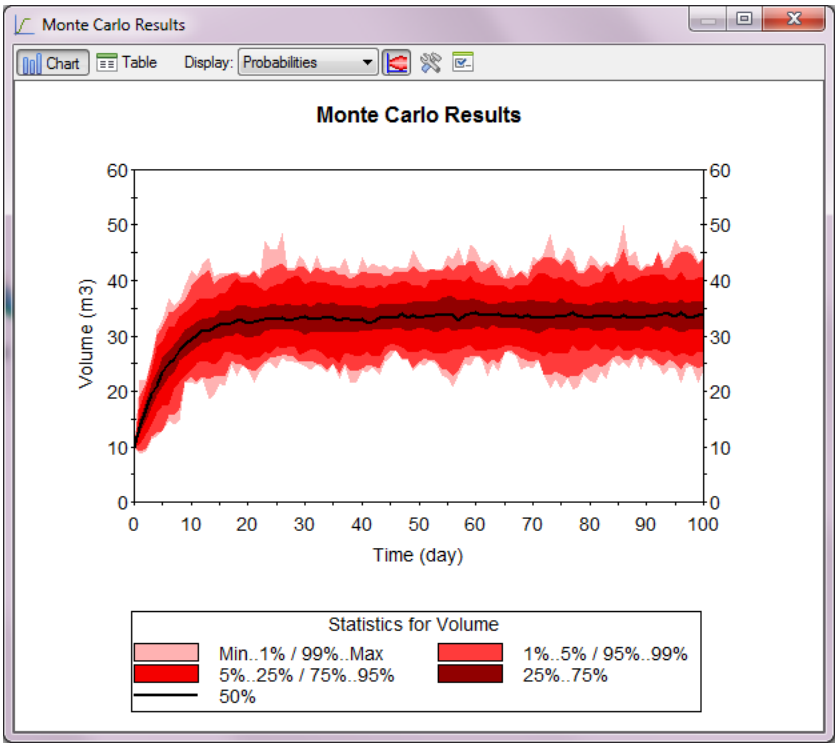
  You can also toggle through the realizations using the **Ctrl+Up** and **Ctrl+Down** keys.

- **All Realizations**: This displays all (unscreened) realizations simultaneously. The chart will show multiple lines. The table displays a column for each realization.

Note: If the Time History Result element contains multiple outputs and/or an array, the display window provides a field for selecting a single result to be displayed when this option is selected.

Read more: Viewing Time Histories of Multiple Realizations for Multiple Outputs (page 561); Viewing Time Histories of Multiple Realizations for an Array (page 562).

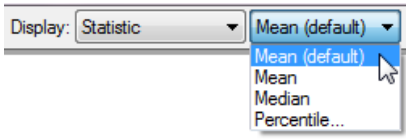
- **Probabilities**: This is the default display. It provides a ***probability histories*** display. This is a powerful probabilistic representation of the time history of an output. In particular, the multiple realizations of the time history are represented by plotting (or tabulating) the statistics (e.g., percentiles, bounds, mean) of the set of time histories:



Read more: Viewing Probability Histories for Multiple Realizations (page 555).

Note: If the Time History Result element contains multiple outputs and/or an array, the display window provides a field for selecting a single result to be displayed when this option is selected.

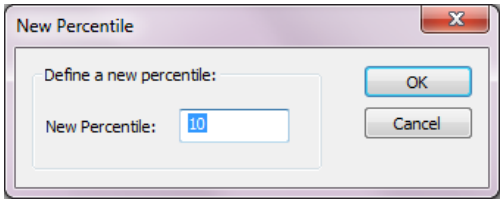
- **Statistic**: This displays a single specified statistic for the collection of (unscreened) realizations. A control is added to the display window to specify the statistic:



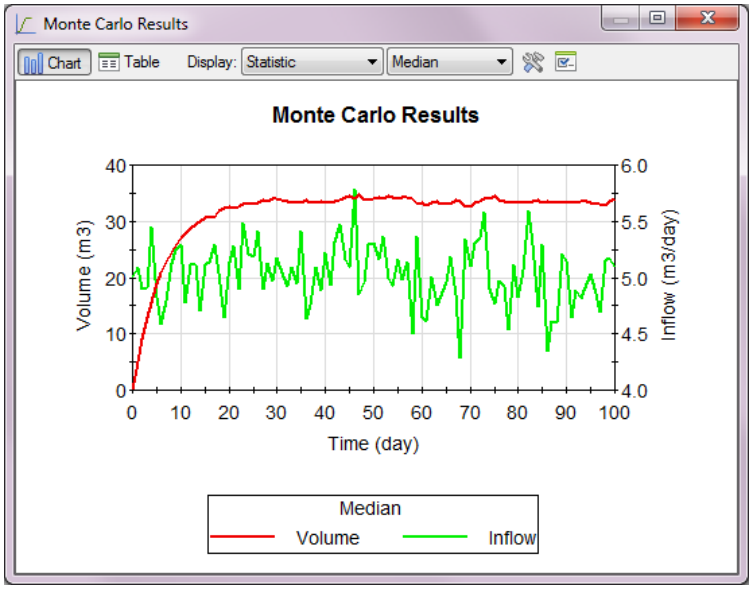
The default statistic that is shown is defined (and can be changed) in the Monte Carlo Result Display dialog (accessed via the **Options…** button in the Result Properties dialog).

Read more: Controlling Monte Carlo Result Options (page 441).

Pressing the **Percentile…** button allows you to define a specific percentile:



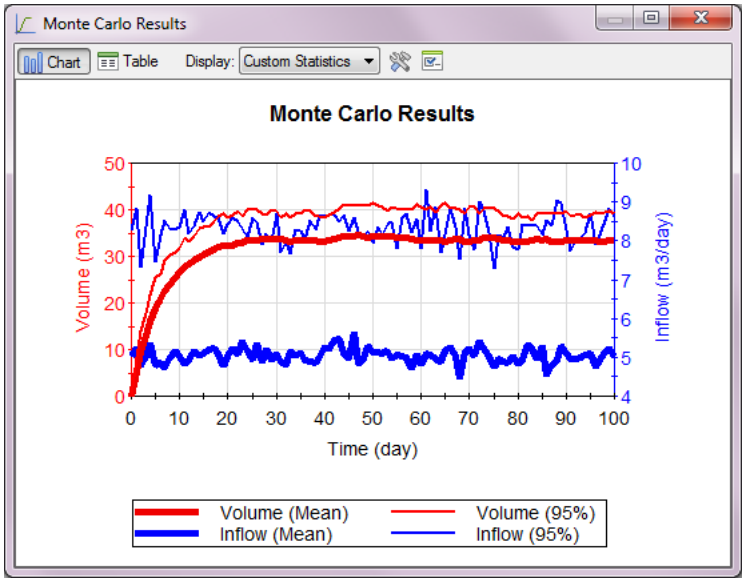
The Statistic option is of particular value when displaying multiple results, as it allows you to quickly view a single statistic for all results:



Note that the legend title indicates the Statistic being plotted.

- **Custom Statistics**: This displays a different custom statistic for each result (as opposed to the same statistic for each result). The Custom Statistic that is shown is specified in the Result Properties dialog.

The Custom Statistic option is valuable if you want to display different statistics for different outputs, or several different statistics for the same output:

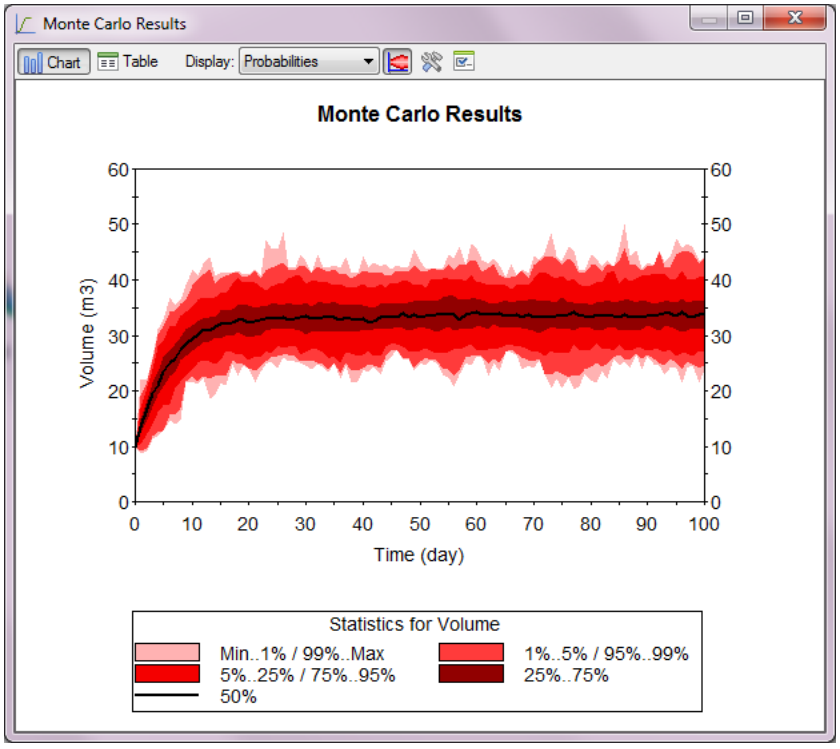


Read more: Viewing Custom Statistics for Multiple Realizations (page 560).

Note: The first time that a Time History Result element is displayed, it will display Probabilities (the default). However, when viewing a Time History Result element, the element "remembers" the last type of display that was selected, and shows that when you double-click on it the next time.

Viewing Probability Histories for Multiple Realizations

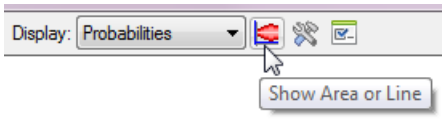
If you have saved multiple realizations for one or more outputs (by referencing them in a time History Result element), by default, a ***probability histories*** display will be shown:

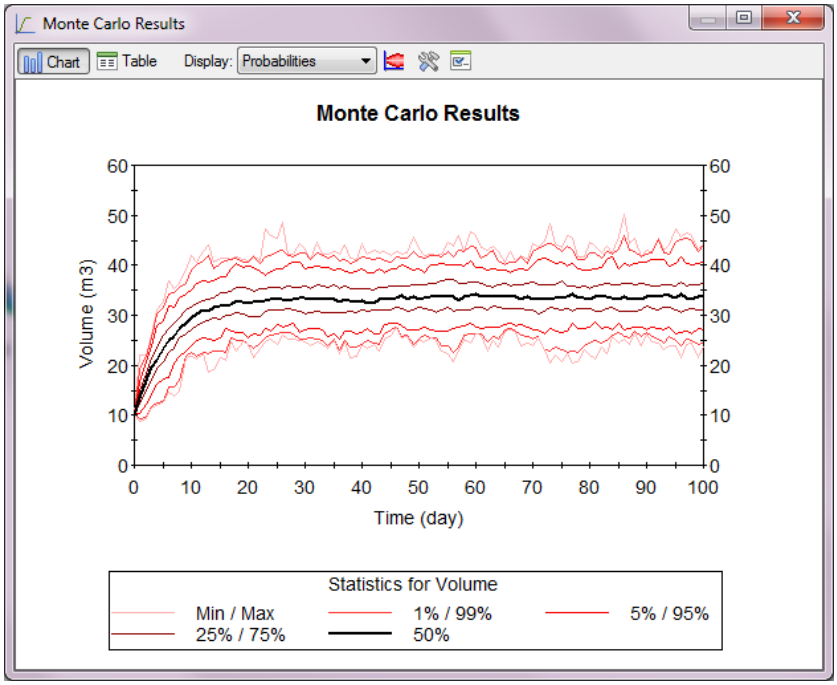


Note: The first time that a Time History Result element is displayed, it will display Probabilities (the default). However, when viewing a Time History Result element, the element "remembers" the last type of display that was selected, and shows that when you double-click on it the next time.

A probability histories display provides a probabilistic representation of the time history of the output. In particular, the multiple realizations of the time history are represented by plotting (or tabulating) the percentiles (as well as the bounds and mean) of the set of time histories for the output.

By default, the areas between the percentile curves are filled. You can show just the lines by clicking the **Show Area or Line** button at the top of the chart:

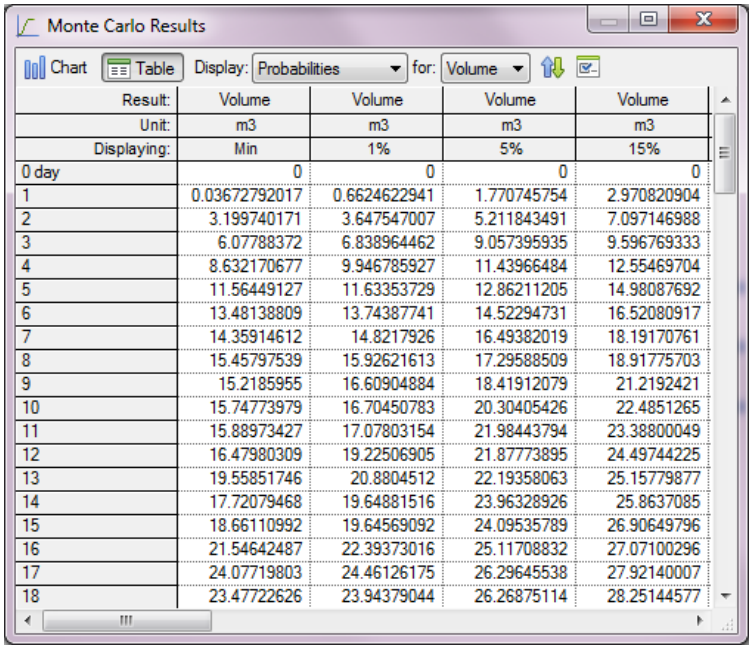




Note: When displaying areas, a context menu option (**Show Outline**) is available. When this is selected, lines are drawn *between* the areas. The line color is the Data Area Foreground color.

Read more: The Chart Style General Tab (page 662).

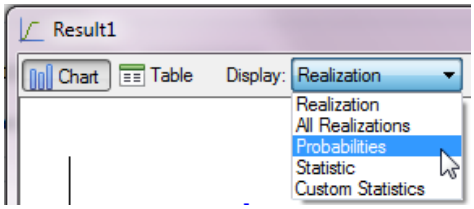
A table looks like this:



Result:	Volume	Volume	Volume	Volume
Unit:	m3	m3	m3	m3
Displaying:	Min	1%	5%	15%
0 day	0	0	0	0
1	0.03672792017	0.6624622941	1.770745754	2.970820904
2	3.199740171	3.647547007	5.211843491	7.097146988
3	6.07788372	6.838964462	9.057395935	9.596769333
4	8.632170677	9.946785927	11.43966484	12.55469704
5	11.56449127	11.63353729	12.86211205	14.98087692
6	13.48138809	13.74387741	14.52294731	16.52080917
7	14.35914612	14.8217926	16.49382019	18.19170761
8	15.45797539	15.92621613	17.29588509	18.91775703
9	15.2185955	16.60904884	18.41912079	21.2192421
10	15.74773979	16.70450783	20.30405426	22.4851265
11	15.88973427	17.07803154	21.98443794	23.38800049
12	16.47980309	19.22506905	21.87773895	24.49744225
13	19.55851746	20.8804512	22.19358063	25.15779877
14	17.72079468	19.64881516	23.96328926	25.8637085
15	18.66110992	19.64569092	24.09535789	26.90649796
16	21.54642487	22.39373016	25.11708832	27.07100296
17	24.07719803	24.46126175	26.29645538	27.92140007
18	23.47722626	23.94379044	26.26875114	28.25144577

Each statistic is displayed in a different column.

If you do not want to display a probability history, you can select a different type of display for multiple realizations from the Display list:



By default, GoldSim plots the following pairs of percentiles when displaying probability histories:

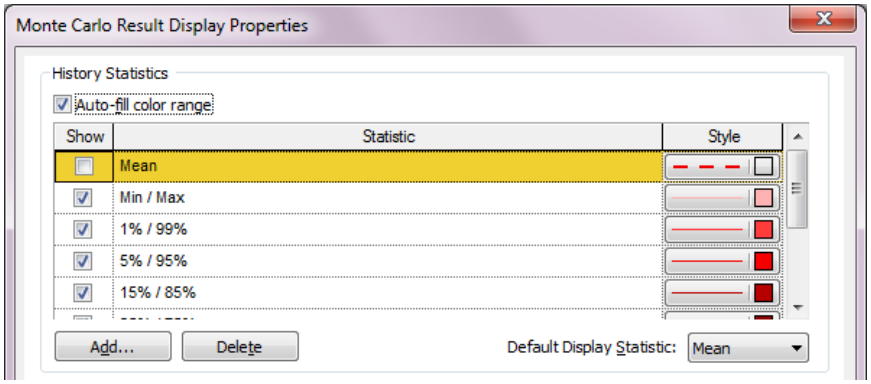
- Min/Max (0%/100%)
- 1%/99%
- 5%/95%
- 15%/85%
- 25%/75%
- 35%/%65
- 45%/%55

It also plots the median (50th percentile). The mean is off by default.

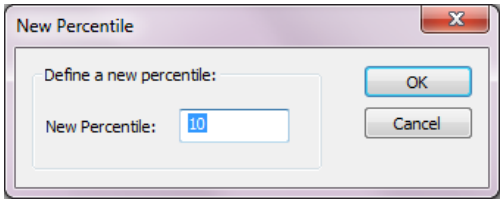
You can customize which of these statistics you would like to plot, as well as add some of your own. You can do so via the Monte Carlo Result Display dialog (accessed via the **Options**… button in the Result Properties dialog).

Read more: Controlling Monte Carlo Result Options (page 441).

The top of this dialog looks like this:



You can activate or deactivate a particular line (or pair of lines) by clicking in the checkbox to the left of the item. You can add new percentile pairs by pressing the **Add…** button. The following dialog is displayed.



You need to only type in one item of a pair (as a percentage, *not* a fraction), and the complementary item will also be added. For example, if you enter 10, both the 10th and 90th percentiles will be added.

You can subsequently remove any percentile pairs using the **Delete** button.

Note: There must be a minimum of two statistics selected: the Mean or Median (50%), and at least one percentile pair.

Warning: You should only view particular percentiles if you have run a sufficient number of realizations such that that statistic can be accurately estimated. As a general rule of thumb, in order to view the Xth percentile (where X is the lower number of the pair, expressed as a fraction), at least 5/X realizations should be available. For example, in order to view the 10th (and 90th) percentile, at least 5/0.1 = 50 realizations should be available. GoldSim does not enforce this limitation automatically (it will try to display any percentiles that you select regardless of the number of realizations available). However, you should manually adjust the percentiles that you choose to view based on the number of realizations available.

Note: The percentiles are truncated at the actual least and greatest values for each timestep. In the absence of importance sampling, these will correspond to probability levels of 1/2N and 1- 1/2N, respectively. For example, if you run 10 realizations, any percentile line below the 5th percentile will be coincident with the 5th percentile.

For each statistic, you select a **Style** for the line and fill color used for the chart. By default, the **Auto-fill color range** option is checked. This option makes it easy to quickly create color gradients for the Styles. Whenever you select one of the **Styles** (and change a color), GoldSim auto-generates colors for the other statistics to create a logical gradient (e.g., if you select red, all the other statistics will be changed to shades of red). If you want to select your own colors for each statistic, you must clear the **Auto-fill color range** option.

Note: Autofill does not apply to the Mean and the Median (50%). These can be edited separately without affecting the color gradient.

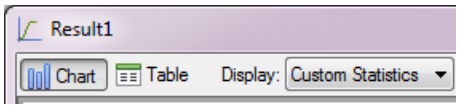
The following additional points should be noted regarding displaying probability histories:

- When you select Mean, GoldSim will also display the Standard Deviation (SD) in tables. However, the SD does not appear in charts.
- All of the information specified in the Monte Carlo Result Display dialog (i.e., the percentiles to display and the styles) is saved with the model and are applied to all probability history displays in the model.
- If the Time History Result element contains multiple outputs and/or an array, the display window provides a field for selecting a single result to be displayed when probability histories are selected.

Read more: Viewing Time Histories of Multiple Realizations for Multiple Outputs (page 561); Viewing Time Histories of Multiple Realizations for an Array (page 562).

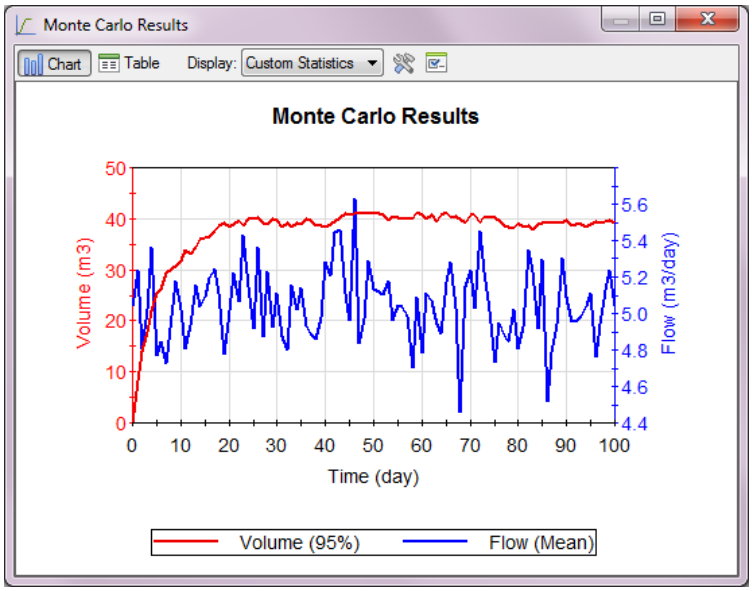
Viewing Custom Statistics for Multiple Realizations

If you have saved multiple realizations for one or more outputs (by referencing them in a time History Result element) one of the display options is to show **Custom Statistics**:



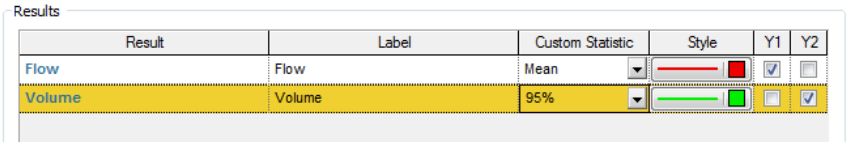
Note: The first time that a Time History Result element is displayed, it will display Probabilities (the default). However, when viewing a Time History Result element, the element "remembers" the last type of display that was selected, and shows that when you double-click on it the next time.

This option displays a different custom statistic for each result:

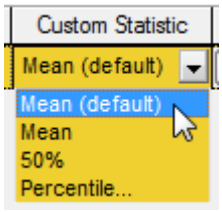


Note that the legend displays the result and the statistic being displayed.

The Custom Statistic that is shown for a particular result is specified in the Result Properties dialog:



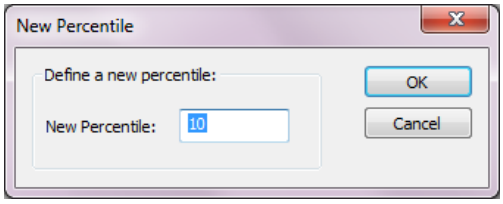
There is a default custom static provided in the list:



The default statistic that is shown is defined (and can be changed) in the Monte Carlo Result Display dialog (accessed via the **Options…** button in the Result Properties dialog).

Read more: Controlling Monte Carlo Result Options (page 441).

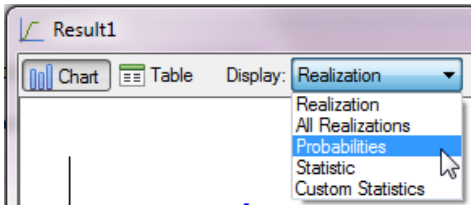
Pressing the **Percentile…** button allows you to define a specific percentile:



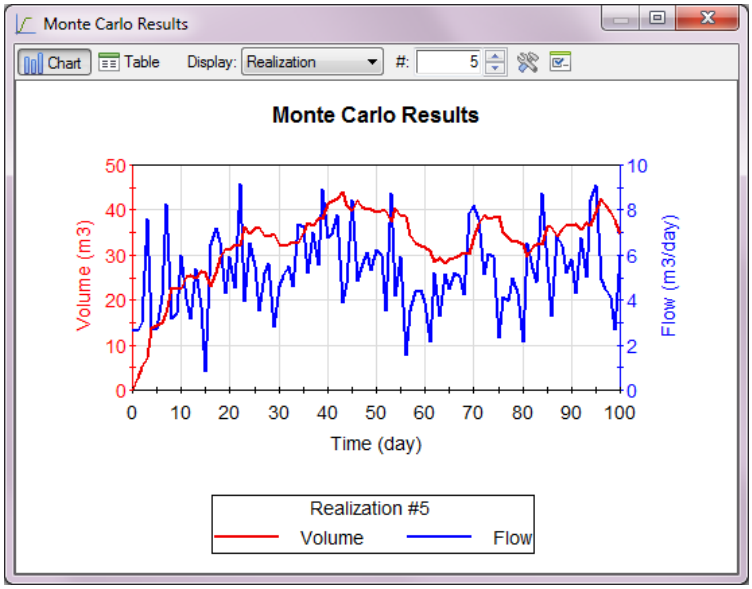
The Custom Statistic option is valuable if you want to display different statistics for different outputs, or several different statistics for the same output.

Viewing Time Histories of Multiple Realizations for Multiple Outputs

If you choose to display a time history of multiple realizations for multiple outputs, what is shown is a function of what has been selected from the Display list:

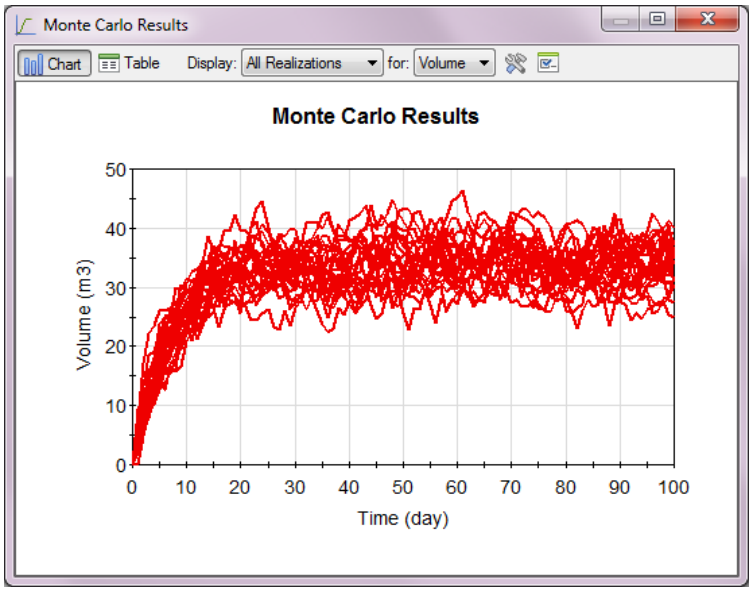


In particular, if **Realization**, **Statistic** or **Custom Statistics** is selected, GoldSim shows the selected result for each output in the list:



This is possible because in these three cases, there is only one set of results to display for each output (e.g., the selected realization for each output).

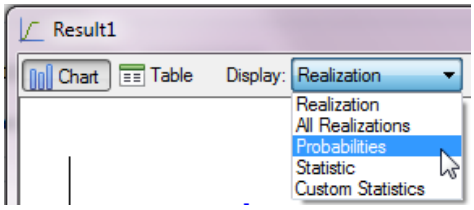
However, if **All Realizations** or **Probabilities** is selected, only one output (result) can be viewed at a time, and it is necessary to select which output you wish to view, since these two displays require multiple sets of results for each output (e.g., all realizations for a selected output):



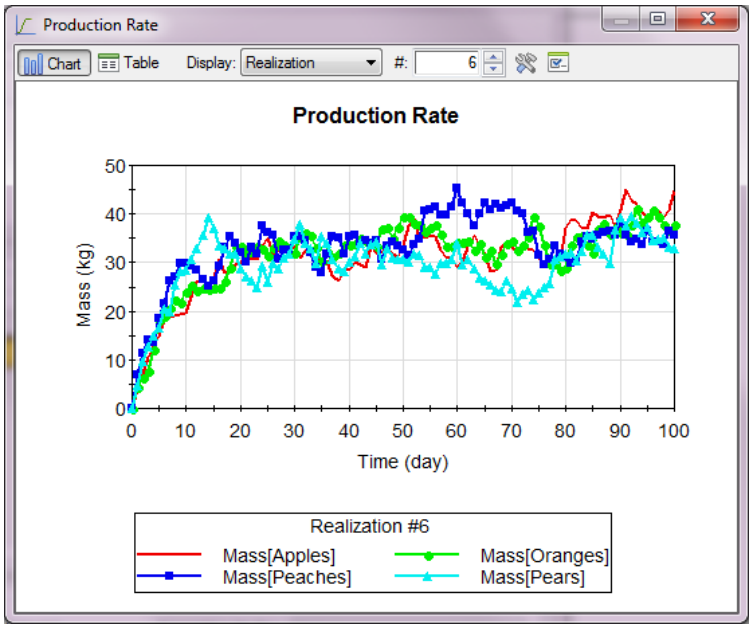
The result that is displayed is selected directly to the right of the **Display** list (after "**for**"). In the example above, the result named "Volume" has been selected.

Viewing Time Histories of Multiple Realizations for an Array

If you choose to display a time history of multiple realizations for an array, what is shown is a function of what has been selected from the Display list:



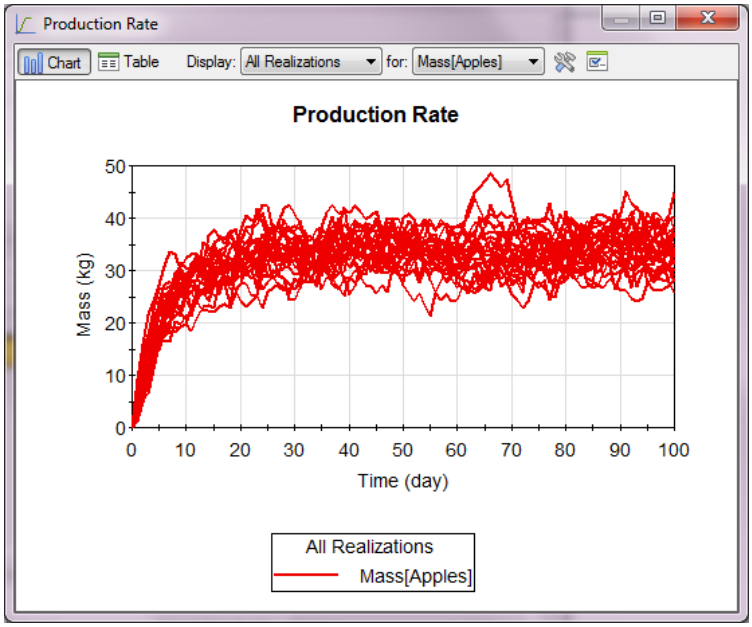
In particular, if **Realization**, **Statistic** or **Custom Statistics** is selected, GoldSim shows the selected result for each item of the array:



In this example, the vector Sales consists of 4 items (Apples, Peaches, Oranges and Pears).

This is possible because in these three cases, there is only one set of results to display for each array item (e.g., the selected realization for each item).

However, if **All Realizations** or **Probabilities** is selected, only one array item can be viewed at a time, and it is necessary to select which item you wish to view, since these two displays require multiple sets of results for each item (e.g., all realizations for a selected item):



The item that is displayed is selected directly to the right of the **Display** list (after "**for**"). In the example above, the result named "Mass[Apples]" (i.e., the item named "Apples" from the output "Mass") has been selected.

Viewing Reporting Period-Based Results in Time History Result Elements

In some cases, you need to display *accumulated*, *average*, the *change* or the *rate of change* of values over specified periods (e.g., monthly, annually). For example, you may need to report the cumulative amount of money or water that moved from one point to another each month.

To support this, GoldSim allows you to define Reporting Periods. Scheduled updates (timesteps) are created at the end of each Reporting Period. Hence, these can also be thought of as "Reporting Steps". However, whereas results saved at Basic Steps always represent instantaneous values, results saved at Reporting Periods can not only be displayed as instantaneous results, but can also report accumulated, averaged, the change or the rate of change of values over a specified period (e.g., monthly).

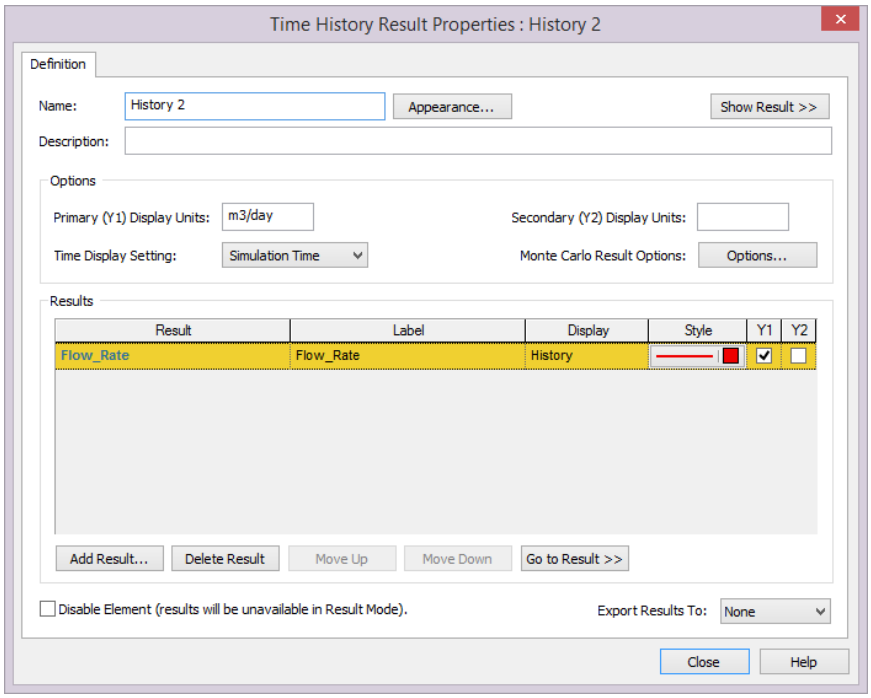
Read more: Defining Reporting Periods (page 421).

Results based on Reporting Periods (e.g., cumulative values over each period, average values over each period, etc.) can be accessed and viewed via Time History Result elements.

It is important to understand that the act of creating Reporting Periods does not in itself provide specialized Reporting Period-based results (e.g., accumulated or averaged results). This is because Reporting Periods by default simply provide another way to define timesteps, since as noted above, scheduled updates (timesteps) are created at the end of each Reporting Period. By default, Time History results simply provide instantaneous values at each timestep (and hence, if Reporting Periods are defined, at the end of each period).

To display specialized Reporting Period-based results, you must add the outputs you are interested in to a Time History Result element, and then specify that you want to view Reporting Period-based results.

By default, the dialog for a Time History Result element looks like this:

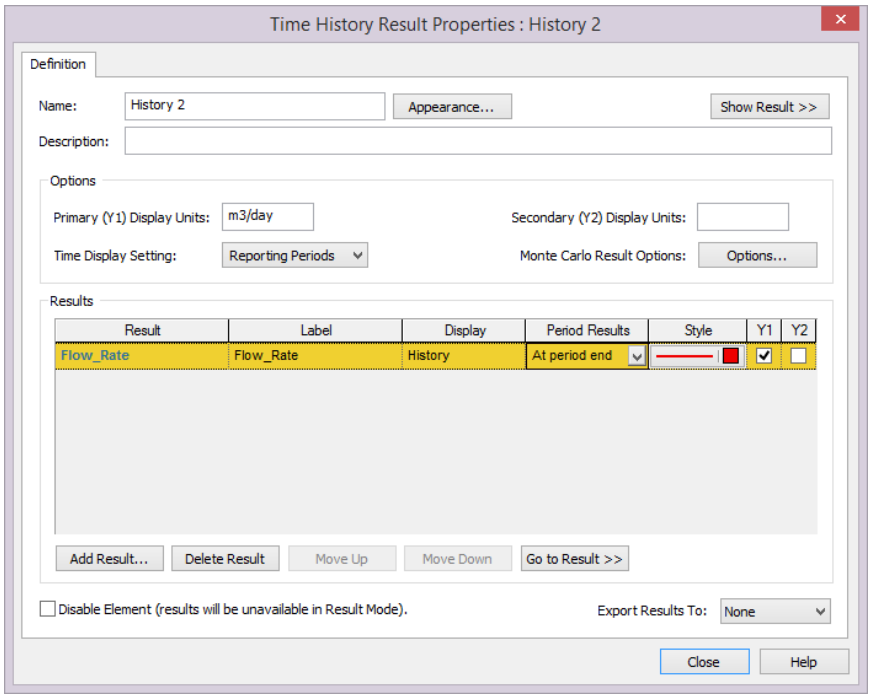


Note: For multiple realization runs, the **Display** column is replaced with **Custom Statistic**.

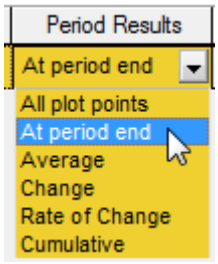
Read more: Viewing Time Histories of Multiple Realizations (page 552).

By default, the **Time Display Setting** in this dialog is set to "Simulation Time". In this case, Time History results simply provide instantaneous values at each timestep (and hence, if Reporting Periods are defined, at the end of each period).

If you have defined Reporting Periods, you can select "Reporting Periods" for the **Time Display Setting**. When you do so, the Result Properties dialog looks like this:



Note that a new column has been added, titled **Period Results**. This is a drop-list that allows you to select results based on Reporting Periods. The list consists the following options:



- **All plot points**: This displays instantaneous values at all the plot points (scheduled updates). This would include not only values at the end of each Reporting Period, but also values at all other scheduled updates (e.g., Basic Steps). This display is identical to what you would see if you selected "Simulation Time" for the **Time Display Settings**. Note, however, that the values of all plot points only appear in charts. In tables, only values at the end of each Reporting Period are shown.
- **At period end**: This displays instantaneous values at at the end of each Reporting Period. It is the default.

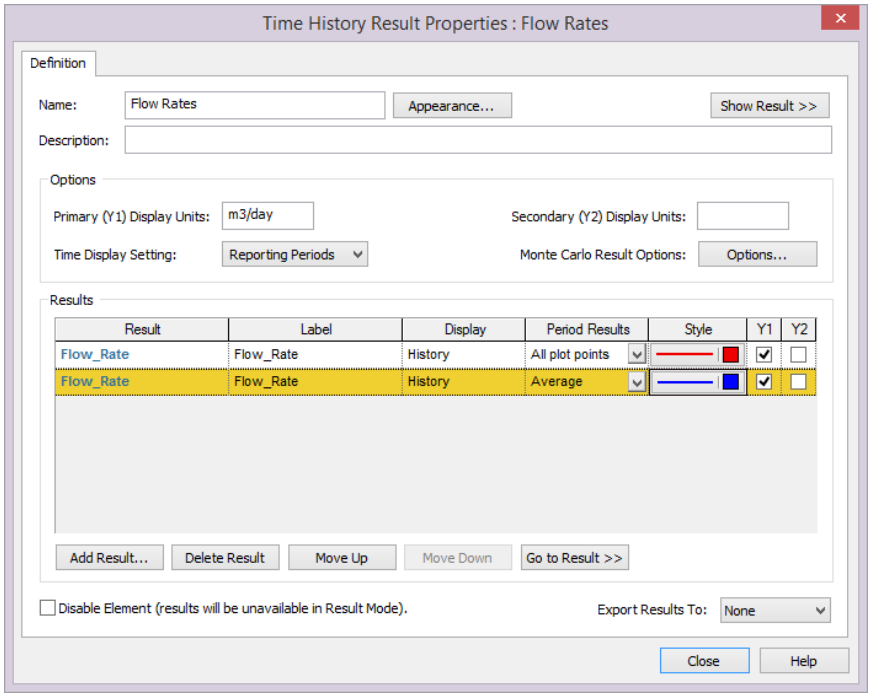
- **Average**: This computes the average value for each Reporting Period. For example, if you had a Basic Step of 1 day, a monthly minor Reporting Period and annual major Reporting Period, GoldSim would display either monthly or annual averages (you select in the display window which to show).
- **Change**: This computes the change in the value over each Reporting Period (the difference between the value at the end of the current Reporting Period and the value at the end of the previous Reporting Period).
- **Rate of Change**: This computes the rate of change in the value over each Reporting Period (from the end of the previous Reporting Period to the end of the current Reporting Period). That is, it is the Change (defined above) divided by the length of the Period. It has dimensions of X/Time, where X is the dimensions of the actual output.
- **Cumulative**: This computes the integral of the value over each Reporting Period (from the end of the previous Reporting Period to the end of the current Reporting Period). It has dimensions of X*Time, where X is the dimensions of the actual output. This would typically be used in cases where the output was a rate (e.g., m^3/day), such that the cumulative value represents the quantity (e.g., m^3) that flowed or accumulated over the period.

Note: Most spreadsheet models implicitly compute *accumulated* values (e.g., cumulative flows) over a step. Hence, if you want to compare GoldSim with a spreadsheet model, you should use Reporting Periods to define your steps, and view the Cumulative result.

Note: If your result is a condition, only two of the Period Result types are applicable: "All plot points" and "At period end". Hence, these are the only two choices in the drop-list.

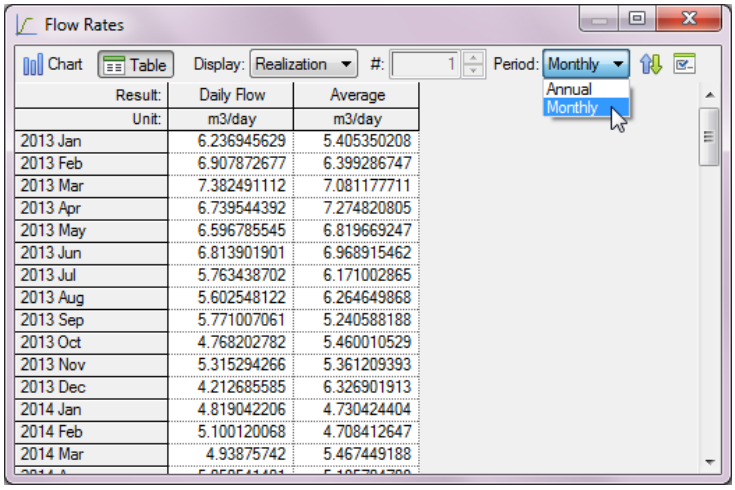
Note: In order to display most types of Reporting Period results, you must add the result to the Result Properties dialog *before* you run the model (i.e., while you are in Edit Mode). This is because they are actually computed "on the fly" as the model is running (e.g., the Cumulative option integrates values during the simulation). Hence, if you try to add a result to a Time History Result element with a **Time Display Setting** of "Reporting Periods" while in Result Mode, the only available Reporting Period options are "All plot points" and "At period end".

To understand the use of Reporting Period-based results, it is instructive to view an example. In the example below, the same input (Flow_Rate) is added to the Result element twice:



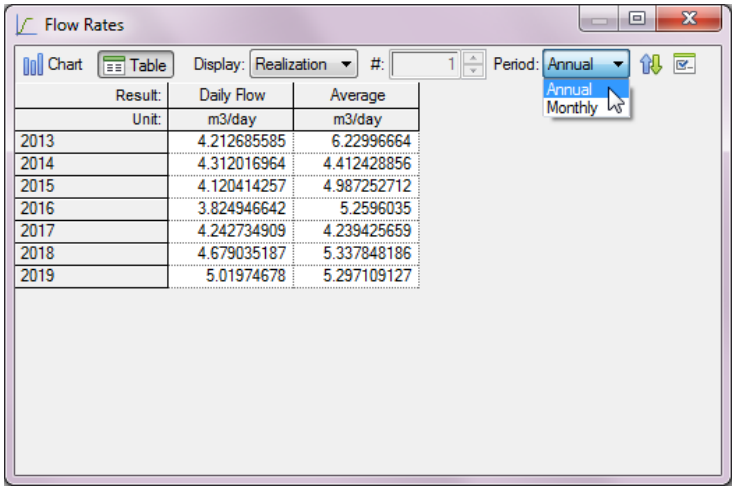
In this particular model, a daily Basic Step has been defined, along with a Major (annual) and minor (monthly) Reporting Period. The first item in the list instructs GoldSim to display the instantaneous value of the flow at all plot points. The second item in the list instructs GoldSim to display the average value of the flow over the Reporting Period.

If you have defined both a Major Period and a Minor Period, you must specify which Period to display when viewing results. In this example, you would need to select whether you wish to display monthly or annual averages at the top of the display. In this case the monthly values are being shown:



Result:	Daily Flow	Average
Unit:	m3/day	m3/day
2013 Jan	6.236945629	5.405350208
2013 Feb	6.907872677	6.399286747
2013 Mar	7.382491112	7.081177711
2013 Apr	6.739544392	7.274820805
2013 May	6.596785545	6.819669247
2013 Jun	6.813901901	6.968915462
2013 Jul	5.763438702	6.171002865
2013 Aug	5.602548122	6.264649868
2013 Sep	5.771007061	5.240588188
2013 Oct	4.768202782	5.460010529
2013 Nov	5.315294266	5.361209393
2013 Dec	4.212685585	6.326901913
2014 Jan	4.819042206	4.730424404
2014 Feb	5.100120068	4.708412647
2014 Mar	4.93875742	5.467449188

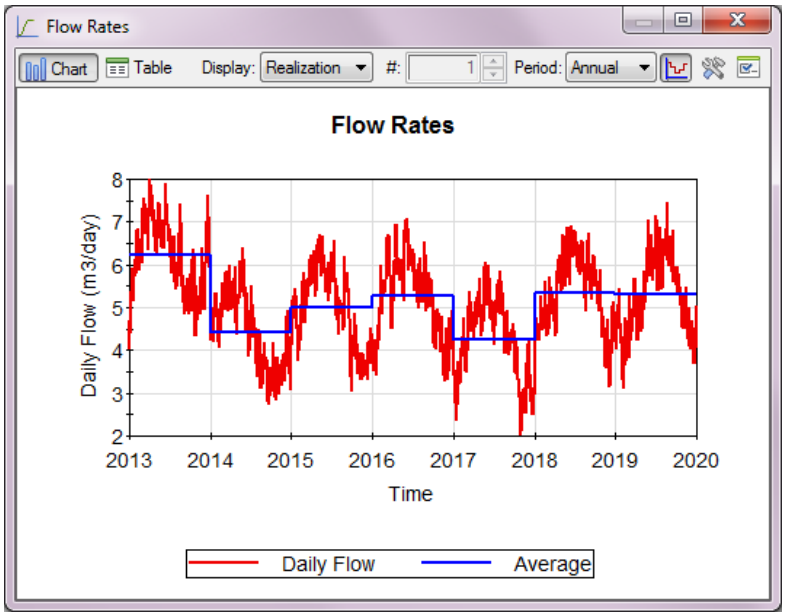
In this case the annual values are being shown:



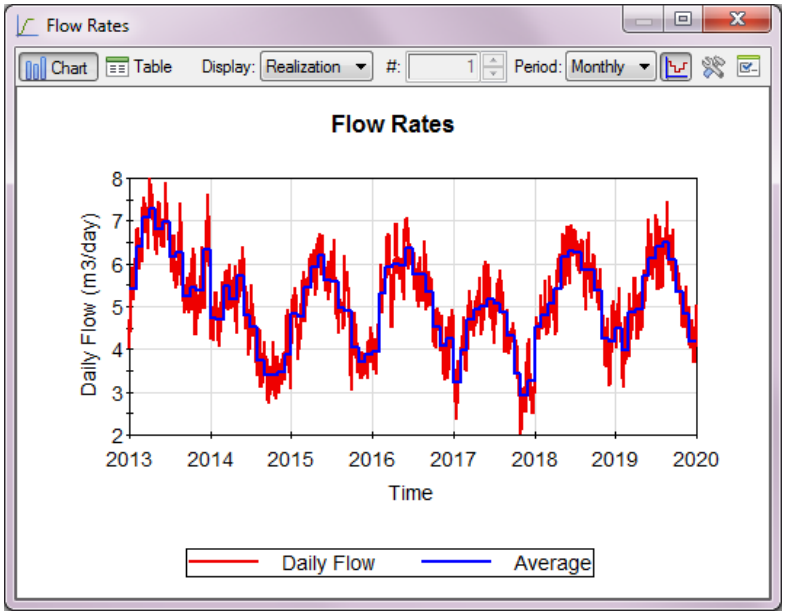
Result:	Daily Flow	Average
Unit:	m3/day	m3/day
2013	4.212685585	6.22996664
2014	4.312016964	4.412428856
2015	4.120414257	4.987252712
2016	3.824946642	5.2596035
2017	4.242734909	4.239425659
2018	4.679035187	5.337848186
2019	5.01974678	5.297109127

Note that the table only displays results corresponding to the actual reporting periods (even though the first item, labeled "Daily Flow" here, was selected to display "All plot points" in the Properties dialog). Hence, in this table, the column labeled "Daily Flow" displays the instantaneous values at the *end* of each period, while the column labeled "Average" displays the average values over the each period. As will be shown below, however, when "All plot points" is selected for an item, all points are displayed in a chart.

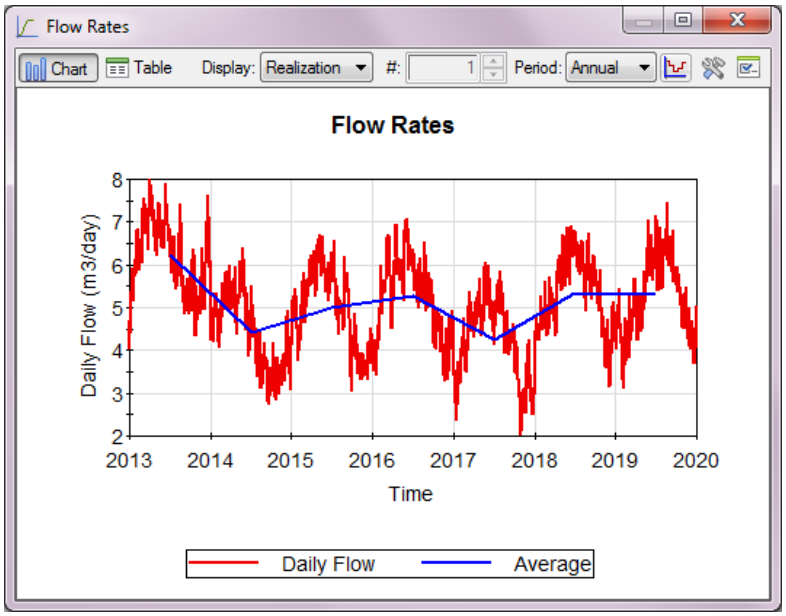
When displaying charts of Reporting Period-based results, you must specify how you would like the values to be plotted. This is because, for four of the result types (Average, Change, Rate of Change and Cumulative), the results actually apply for an *entire period* (as opposed to the end of the period). To represent this, GoldSim provides two options for plotting these results, illustrated in the examples below. In these examples, the Daily Flow and the Average are displayed on the same chart:



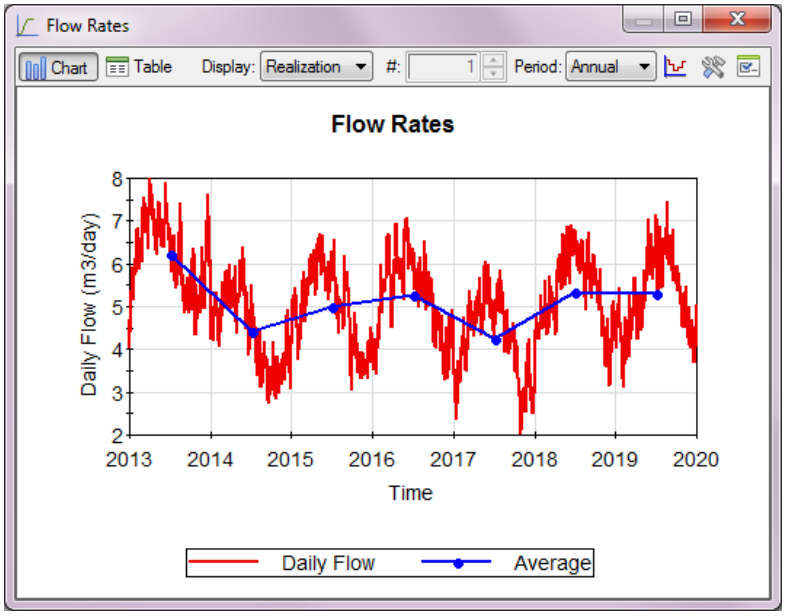
In this case, the major (annual) Reporting Period Average is plotted as a horizontal line over the period (a stair-step). Here is the same plot displaying the minor (monthly) Reporting Period results:



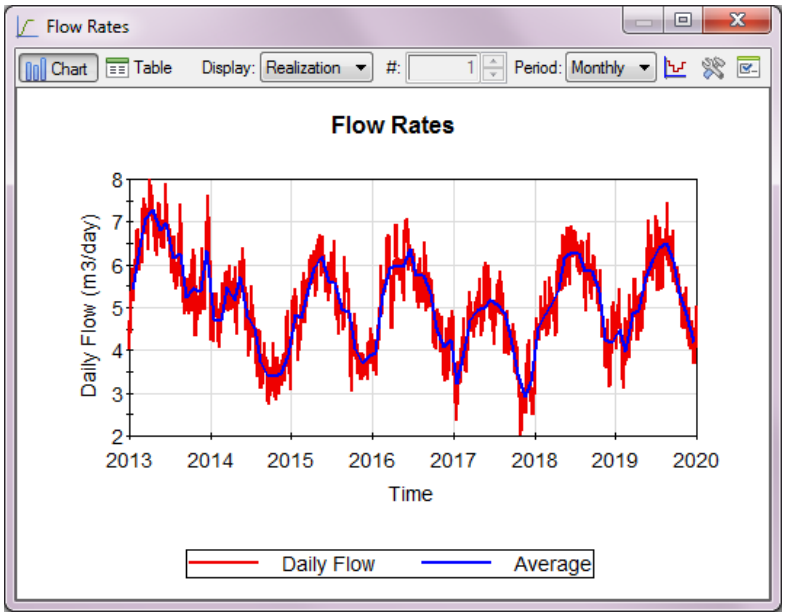
Alternatively, the plot can be displayed like this (these are showing annual results again):



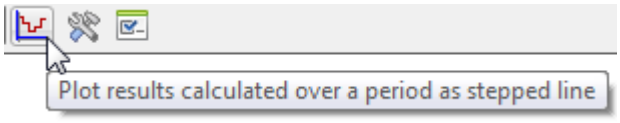
In this case, the major (annual) Reporting Period Average is plotted as a line connecting the **mid-point** of each period (this can be seen better by changing the style to add a symbol at each data point):



Here is the same plot (without the symbols) displaying the minor (monthly) Reporting Period results with lines drawn between mid-points:



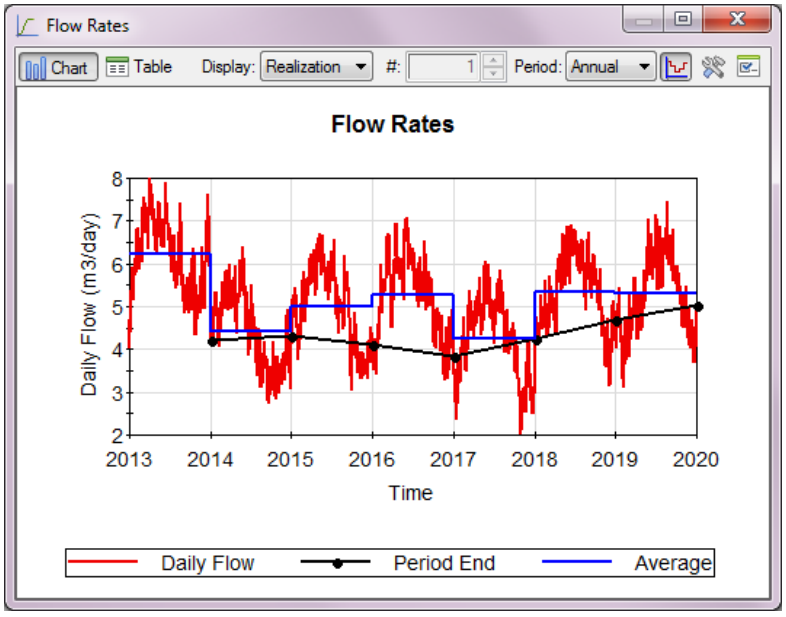
The type of plot is controlled by this button at the top of the display window:



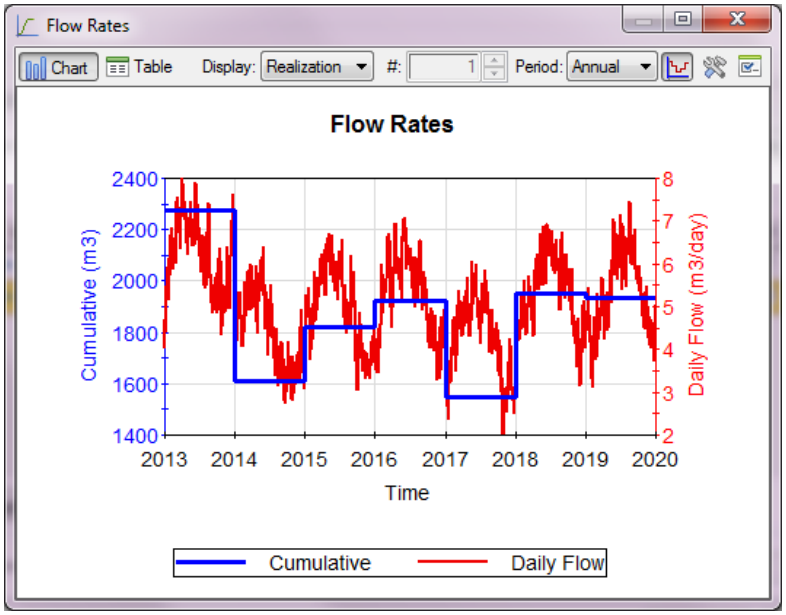
If the button is pressed, the plot is shown as a stair-step; otherwise it is shown as lines connected the mid-points.

Note that this option (to plot as a stair-step) is only used when you have selected to display Average, Change, Rate of Change or Cumulative Reporting Period-based results. If you select "All plot points", all points are plotted (and connected by lines), as shown in the "Daily Flow" line above. If you select "At

period end", the values are plotted at the end of each period (as opposed to the mid-points) plotted (connected by lines):



Note that if you select either "Cumulative" or "Rate of change", and you simultaneously select any of the other result types, the Y2 axis must also be used, since the dimensions of the result are different from that of the output:



A simple example file illustrating Reporting Periods (ReportingPeriods.gsm) can be found in the Timestepping subfolder of the General Examples folder in your GoldSim directory.

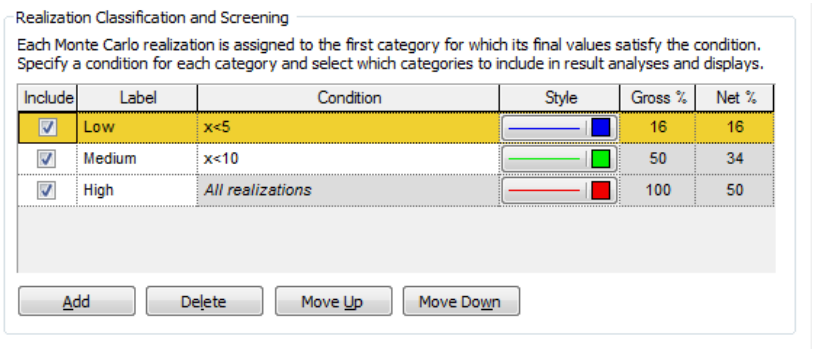
Using Result Classification and Screening in Time History Results

When carrying out probabilistic simulations, you may often run hundreds or thousands of realizations. In order to analyze the results, it is often quite useful to classify the realizations into *categories*. A category is simply defined by a condition relating one or more outputs in the model (e.g., those realizations in which the discount rate was above 3.5%; those realizations in which the profit

exceeded $1,000,000; those realizations in which the peak concentration was between 1 mg/l and 10 mg/l).

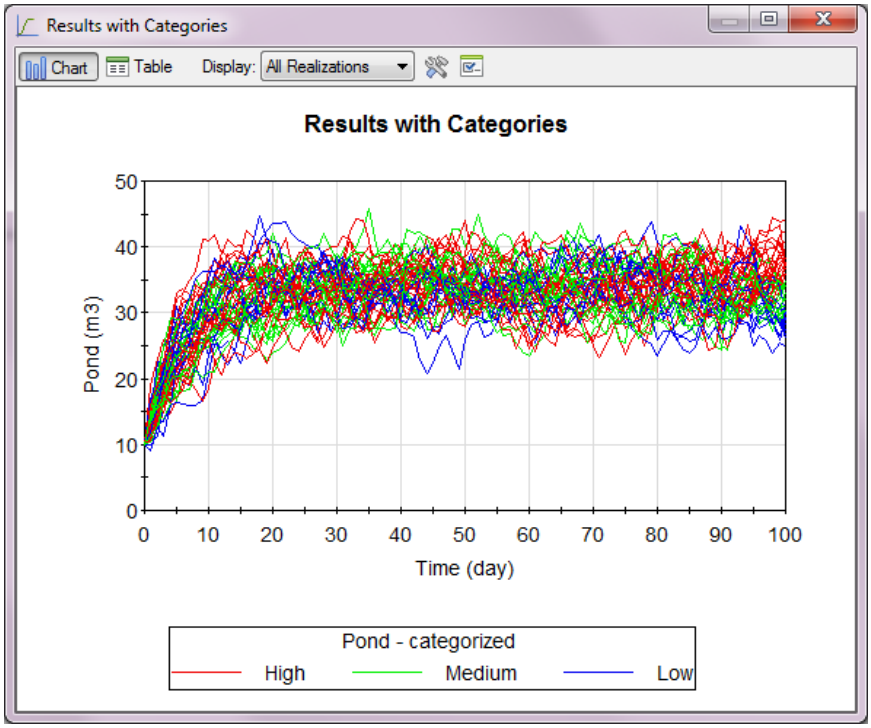
Read more: Classifying and Screening Realizations (page 533).

Classification categories are defined at the bottom of the Monte Carlo Result Display Properties dialog:



This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the **Result Options…** button), and is also accessible by pressing the **Options…** button available in the Result Properties dialog of a Time History Result element.

When viewing a Time History result, if 1) you have run multiple realizations; 2) you have defined more than one category; and 3) you are displaying **All Realizations**, GoldSim will label the curves in the Time History Chart based on the categories you have defined:

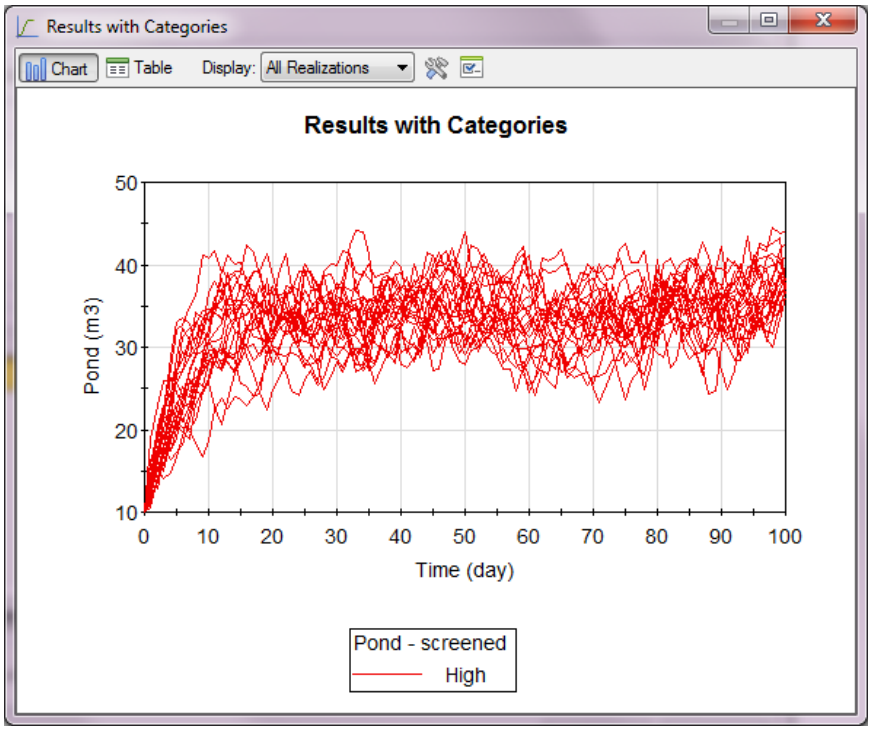


Read more: Viewing Time Histories of Multiple Realizations (page 552).

In the example shown above, three categories have been defined. The chart is displaying all realizations for a single result, with each curve having a style

defined by its category. (Note that for each category, you can edit the **Style** via the Monte Carlo Result Display Properties dialog.)

In addition, screening by category is applied when displaying time histories. That is, if you have chosen to ***screen*** out one or more categories (by clearing the **Include** box in the dialog above), the results (e.g., statistics, realizations) that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include. In the example below, realizations falling into the Low and Medium categories have been screened:

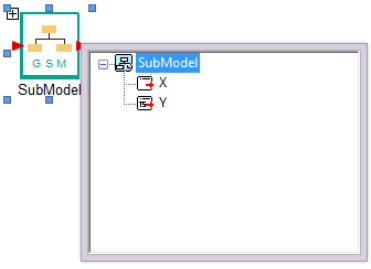


Note that the legend indicates that some categories have been screened out.

Viewing SubModel Results in Time History Result Elements

In most cases, when you add an output to a Time History Result (via the **Add Result…** button in the Result Properties dialog), you will simply be adding a standard output (such as the output of an Expression element or the the primary output of a Reservoir element). However, in one special case, you can add (and view) a specialized output referred to as a Time History Definition output in a Time History Result.

Time History Definition outputs are complex outputs that represent all the information necessary to define a time series. These outputs can be clearly identified when viewing the output port of an element:



In this example, Y is a Time Series Definition Output. Note that the Time History Definition output has a different icon than standard outputs.

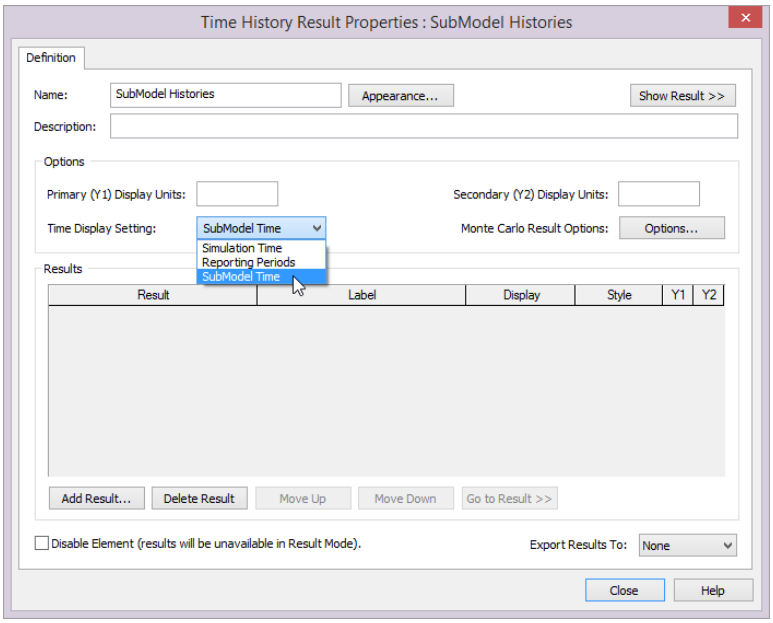
Time Series Definition outputs can be produced by several types of elements, but *only those produced by SubModels can be added to a Time History Result*.

Read more: Using SubModels to Embed Models Within Models (page 914); Creating the Output Interface to a SubModel (page 925).

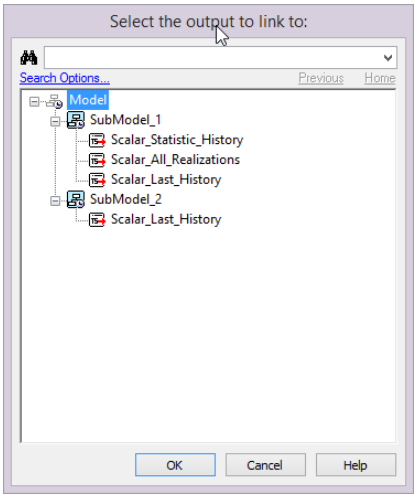
Adding a Time History Definition output of a SubModel to a Time History Result in the parent model provides a mechanism to view the time history results generated by the SubModel directly within the parent model.

In order for SubModel time history results to be displayed in the parent model, you must do the following:

1. Create a Time History Result element in the parent model *before running the model* (i.e., while the model is in Edit Mode).
2. Within the Result Properties dialog, select "SubModel Time" for the **Time Display Setting**:



3. When you do so, this restricts the types of results that can be added via the **Add Result…** button. In particular, when you press this button, you will only be presented with a list of SubModels, and you can only add SubModel Time History Definition outputs:



Several points should be noted:

- You can only add results from a single SubModel. Hence, once you add results from one SubModel, any subsequent results that you add must come from the same SubModel. If you add results from different SubModels, you will not be able to run the model.
- There are three different types of Time History Definition outputs that can be generated by a SubModel (Last Calculated, Statistic History, Realization Histories).

  Read more: Creating the Output Interface to a SubModel (page 925).

  If you want to plot Realization Histories, it is the only result that can be added to the Result element. If you add multiple results, and one of the results consists of Realizations Histories, you will not be able to run the model.
- The time history results associated with a SubModel can (and typically will) have completely different Time settings than the parent model. When viewing time history results associated with a SubModel in the parent model, these are always based on the Time settings within the SubModel.
- Similarly, the results from a SubModel can (and typically will) have completely different Monte Carlo settings than the parent model. When viewing time history results associated with a SubModel in the parent model, probabilistic results will appropriately reflect the Monte Carlo settings of both the SubModel and the parent model.
- Scenario results are not saved for time history results associated with a SubModel. If a Result element in a parent model is linked to SubModel Time History Definition outputs, no results will be displayed in Scenario Mode.

  Read more: Viewing Scenario Results in Time History Result Elements (page 578).

Note: The Time settings of the parent model have no effect on how time history results associated with a SubModel are ploted in a Time History Result element within the parent model.

As pointed out above, when creating a Time History Definition output for a SubModel, three types of outputs can be specified:

Last Calculated: This represents a single time history of the output for the final realization of the SubModel (which is equivalent to the *only* realization if the SubModel Monte Carlo Settings do not specify multiple realizations).

Statistic: This represents a single time history for a specified statistic (computed over all realizations of the SubModel).

Realization Histories. This is the time history of the output for *all* realizations of the SubModel.

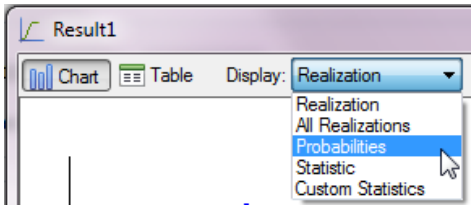
Read more: Creating the Output Interface to a SubModel (page 925).

The options provided for displaying time history results associated with a SubModel in a Time History Result element within the parent model are a function of the type of Time History Definition output being displayed and the Monte Carlo settings of the SubModel and parent model.

Viewing Last Calculated and Statistic History Results from a SubModel

Due to the manner in which they are defined, "Last Calculated" and "Statistic" Time History Definition outputs produce a single time history for each realization of the parent model (regardless of how many realizations were carried out for the SubModel).

Therefore, the manner in which these results are viewed is identical to how time histories of multiple realizations would be viewed for any other type of output. In particular, when you are viewing time histories of Last Calculated or Statistics for a SubModel output after you have run multiple realizations of the parent model, there are a number of different ways that you can view the results (e.g., view one realization at a time, view all realizations, view a particular statistic). The Display windows provide a number of options that allow you to select how you want to view the results:



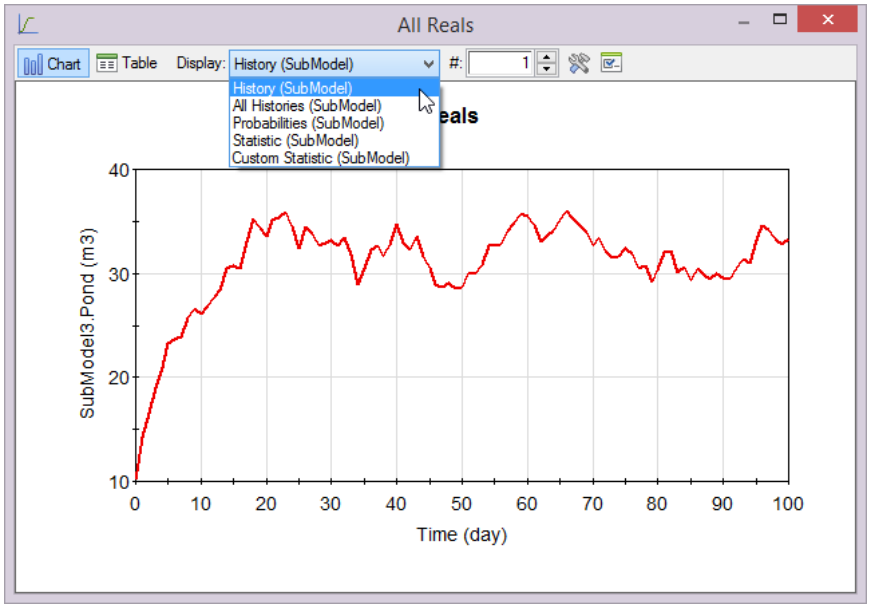
Read more: Viewing Time Histories of Multiple Realizations (page 552).

Note: If only one realization of the parent model is run, only one option (Display Realization/History) will be available.

Viewing Realization Histories Results from a SubModel

Viewing "Realization Histories" outputs from a SubModel is considerably more compex than viewing "Last Calculated" and "Statistic" outputs, since "Realization Histories" Time History Definition outputs can produce *multiple* time histories for each realization of the parent model. This is because "Realization Histories" consist of time histories of *all* realizations of the SubModel. For example, if the parent model was run for 100 realizations, and the SubModel was run for 50 realizations, the results being displayed in a Time History Result element within the parent model would be based on 5000 individual time histories (50 * 100).

Fortunately, GoldSim provides some powerful tools to view this complex set of results. To better understand these tools, it is instructive to first understand how such results would be viewed in a simpler case: a model in which the SubModel was run for multiple realizations, but the parent model was run only for a single realization. In such a case, the Result Display (for a chart) would look like this:

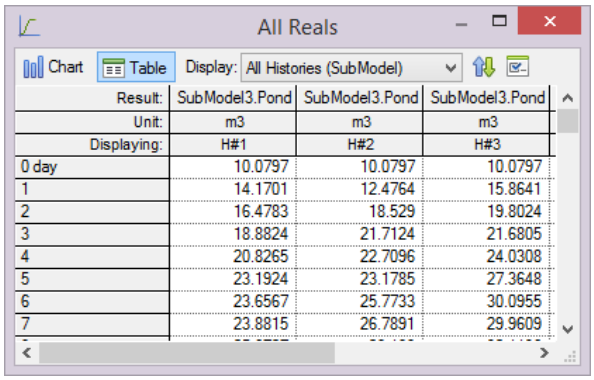


In this particular case, the manner in which these results are viewed is identical to how time histories of multiple realizations would be viewed for any other type of output. The Display windows provide a number of options that allow you to select how you want to view the results.

Read more: Viewing Time Histories of Multiple Realizations (page 552).

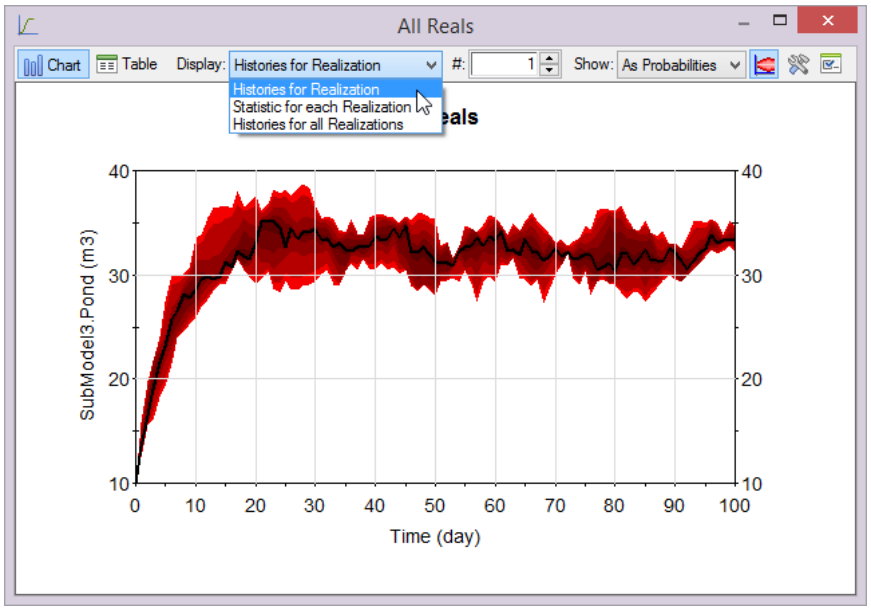
Note: If only one realization of the SubModel is run, only one option (Display History) will be available.

However, there is one very important distinction that is critical to note. This result is not displaying multiple realizations of the parent model (recall that only a single parent model realization was carried out). This result is displaying multiple realizations *from within the SubModel*. That is, although the Result element is located in the parent model, in this particular case, it is displaying the realizations carried out by the SubModel. This is clearly indicated in the Display drop-list by listing "(SubModel)" after each of the five display options. This is also indicated when displaying tables in the manner that the histories are labeled:



Note that that the various time histories are labeled as H#1, H#2, and so on. For realizations of the parent model, they would be labeled as R#1, R#2, etc. The idea is that H refers to a SubModel "history", while R refers to a parent model "realization".

Result display is more complex when both the parent model and the SubModel are run for multiple realizations (referred to as "nested Monte Carlo simulation"):



As can be seen, multiple options are provided to allow you to view the nested Monte Carlo results. These options are discussed in detail in the section discussing nested Monte Carlo simulation listed below.

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933).

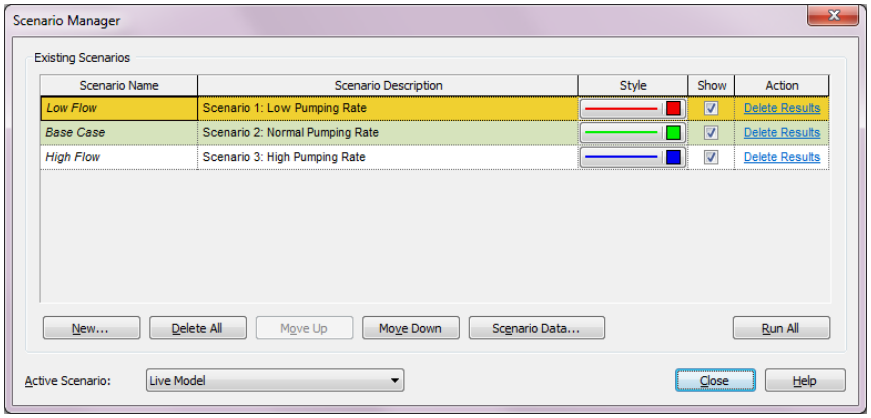
Viewing Scenario Results in Time History Result Elements

GoldSim's scenario modeling capability allows you to directly compare results generated by different sets of input parameters. In effect, when you use this capability, your model can store (and subsequently compare) multiple sets of results (and inputs).

Read more: Creating, Running and Comparing Scenarios (page 463).

When a model is in Scenario Mode, Time History Result elements can be used to view scenario results.

In order for scenario results to be displayed in a Time History Result element, the **Show** button must be checked for each scenario to be displayed in the Scenario Manager (accessed from the main menu (**Run | Scenario Manager…**) or by pressing **F7**):



In this example, the Show box is checked for all three scenarios, so all three will be displayed in results.

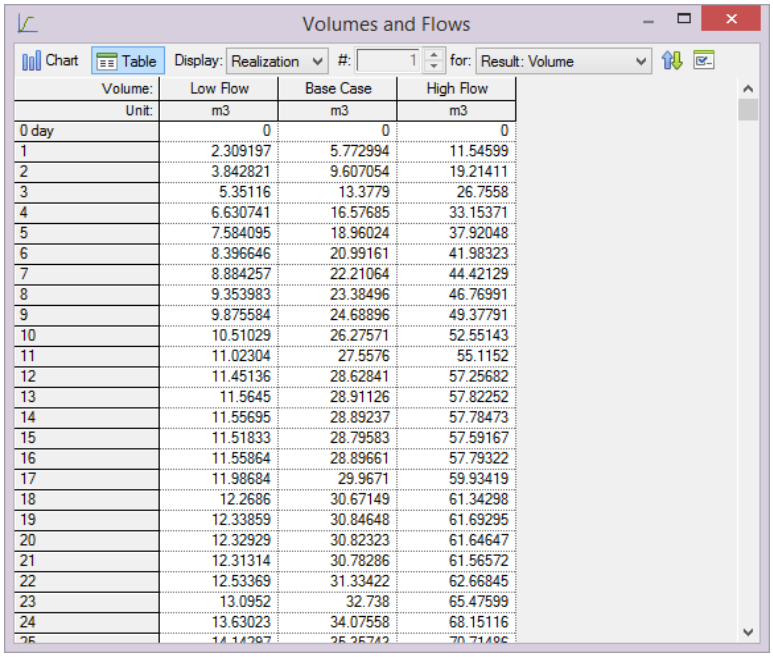
If you double-click on a Time History Result element in Scenario Mode, GoldSim displays results for all scenarios for which scenario results have been generated (and for which the **Show** button has been checked from within the Scenario Manager).

Note that for each scenario, within the Scenario Manager you can edit the **Style**, as well as the **Scenario Name**. These affect how the results are labled in chart and table displays.

Read more: Controlling the Chart Style in Time History Results (page 592).

The time history display for scenarios differs depending on how the simulation was run (single realization or multiple realizations), what you choose to display (e.g., a statistic or all realizations), and whether the Time History Result element contains multiple results.

In all cases, you can select which results and which scenarios you wish to display. For example, if only a single realization was run (for three different scenarios), the Table display in Scenario Mode would look like this:

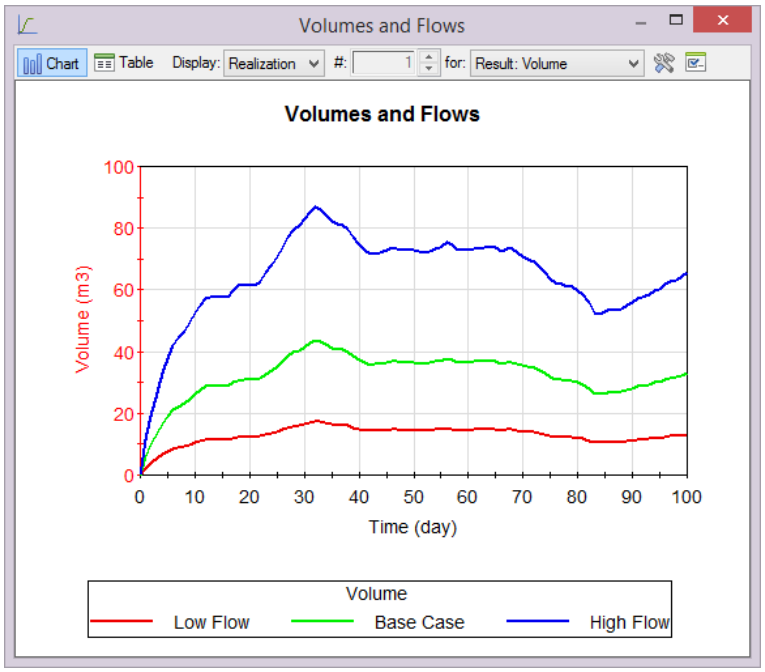


Volume:	Low Flow	Base Case	High Flow
Unit:	m3	m3	m3
0 day	0	0	0
1	2.309197	5.772994	11.54599
2	3.842821	9.607054	19.21411
3	5.35116	13.3779	26.7558
4	6.630741	16.57685	33.15371
5	7.584095	18.96024	37.92048
6	8.396646	20.99161	41.98323
7	8.884257	22.21064	44.42129
8	9.353983	23.38496	46.76991
9	9.875584	24.68896	49.37791
10	10.51029	26.27571	52.55143
11	11.02304	27.5576	55.1152
12	11.45136	28.62841	57.25682
13	11.5645	28.91126	57.82252
14	11.55695	28.89237	57.78473
15	11.51833	28.79583	57.59167
16	11.55864	28.89661	57.79322
17	11.98684	29.9671	59.93419
18	12.2686	30.67149	61.34298
19	12.33859	30.84648	61.69295
20	12.32929	30.82323	61.64647
21	12.31314	30.78286	61.56572
22	12.53369	31.33422	62.66845
23	13.0952	32.738	65.47599
24	13.63023	34.07558	68.15116

Read more: Viewing a Time History Table (page 542).

By default, all scenarios are displayed for the first result listed in the Properties dialog for the Result element. Each column then represents a scenario. The result being displayed is indicated in the upper left-hand corner of the table.

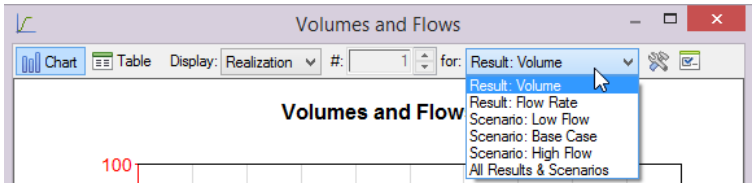
The corresponding chart in Scenario Mode would look like this:



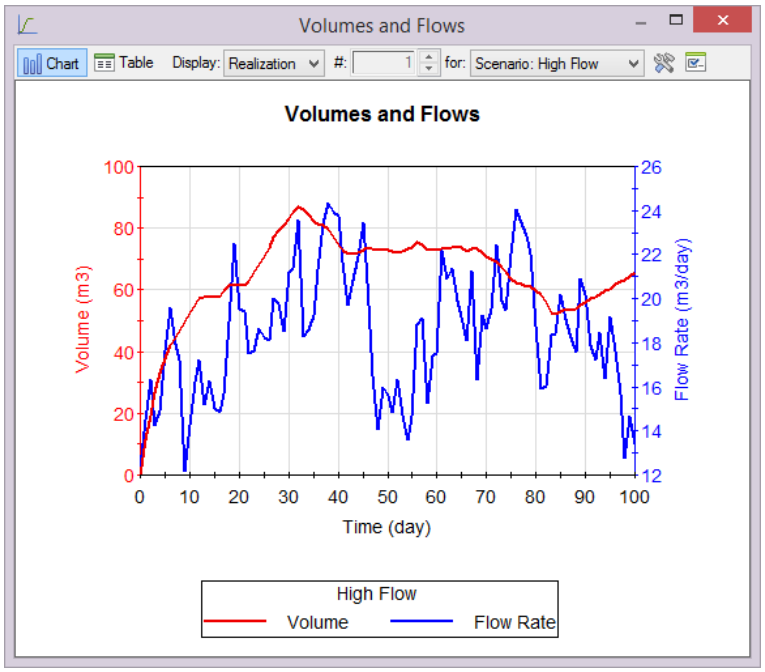
Read more: Viewing a Time History Chart (page 540).

The Scenario Names are shown in the legend. The Result being displayed is indicated in the legend title.

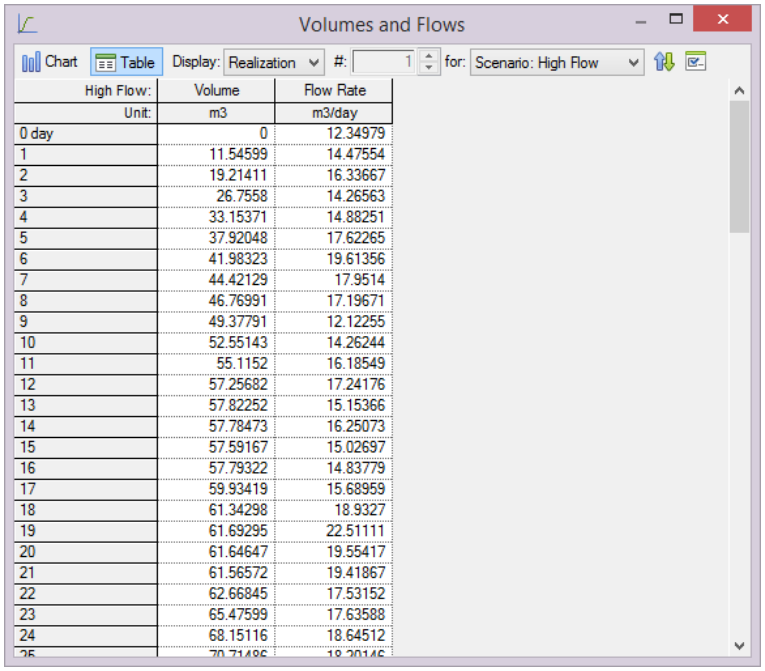
You can specify exactly which scenarios and results you wish to display using a drop-list at the top of the display window. In the example below, there are two results and three scenarios:



Selecting an option prefaced with "Result" displays all the scenarios for that particular result (as shown in the table and chart shown above). Selecting an option prefaced with "Scenario" displays all the results for that particular scenario (as illustrated below):



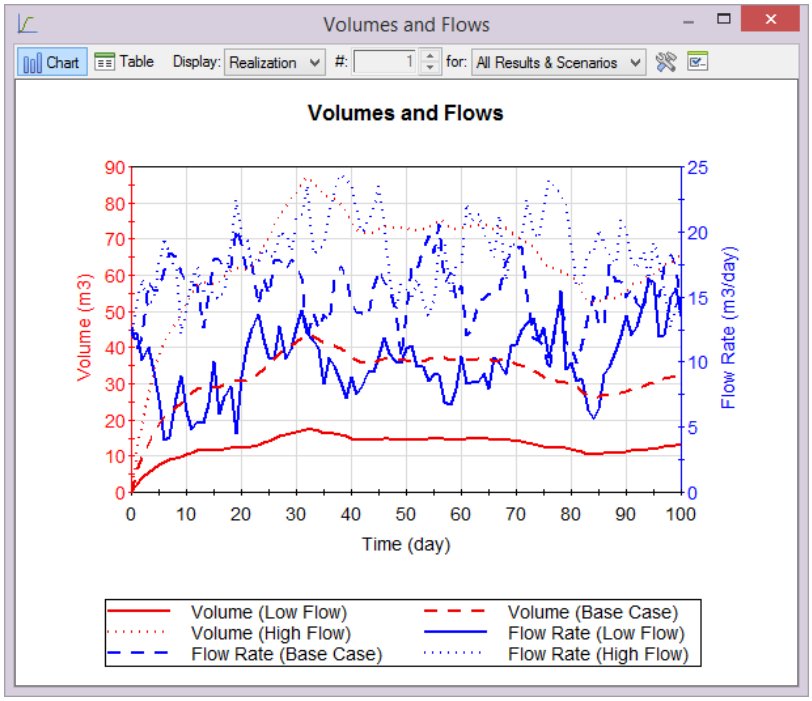
In this example, two different results are shown (Volume and Flow Rate) for the selected scenario (High Flow). The corresponding Table display would look like this:



High Flow:	Volume	Flow Rate
Unit:	m3	m3/day
0 day	0	12.34979
1	11.54599	14.47554
2	19.21411	16.33667
3	26.7558	14.26563
4	33.15371	14.88251
5	37.92048	17.62265
6	41.98323	19.61356
7	44.42129	17.9514
8	46.76991	17.19671
9	49.37791	12.12255
10	52.55143	14.26244
11	55.1152	16.18549
12	57.25682	17.24176
13	57.82252	15.15366
14	57.78473	16.25073
15	57.59167	15.02697
16	57.79322	14.83779
17	59.93419	15.68959
18	61.34298	18.9327
19	61.69295	22.51111
20	61.64647	19.55417
21	61.56572	19.41867
22	62.66845	17.53152
23	65.47599	17.63588
24	68.15116	18.64512

Each column represents a result. The scenario being displayed is indicated in the upper left-hand corner of the table.

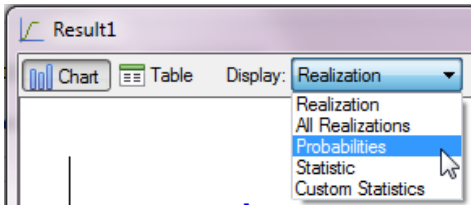
There is also an option to display “All Results & Scenarios”:



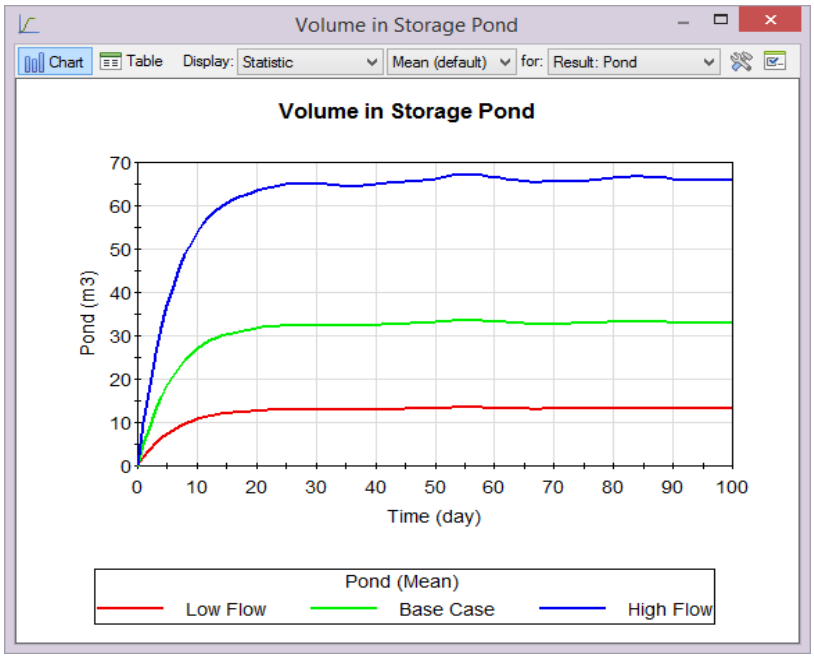
Note: Array results cannot be compared in Scenario Mode. Only scalar results can be displayed. Hence, if an array is listed as one of the results in the Result Properties page, it will not be available for selection in the display dialogs when comparing scenarios (i.e., when the model is in Scenario Mode). If you wanted to compare one or more items from an array in Scenario Mode, you would need to add those scalar array items as separate results in the Result Properties dialog.

Note: Scenario results are only available for results that are added to the Result element before you run a scenario. If you add a result after you run a scenario, scenario results for that output will not be available for that scenario.

If you have run multiple realizations for the various scenarios, what is shown is a function of what has been selected from the Display list:

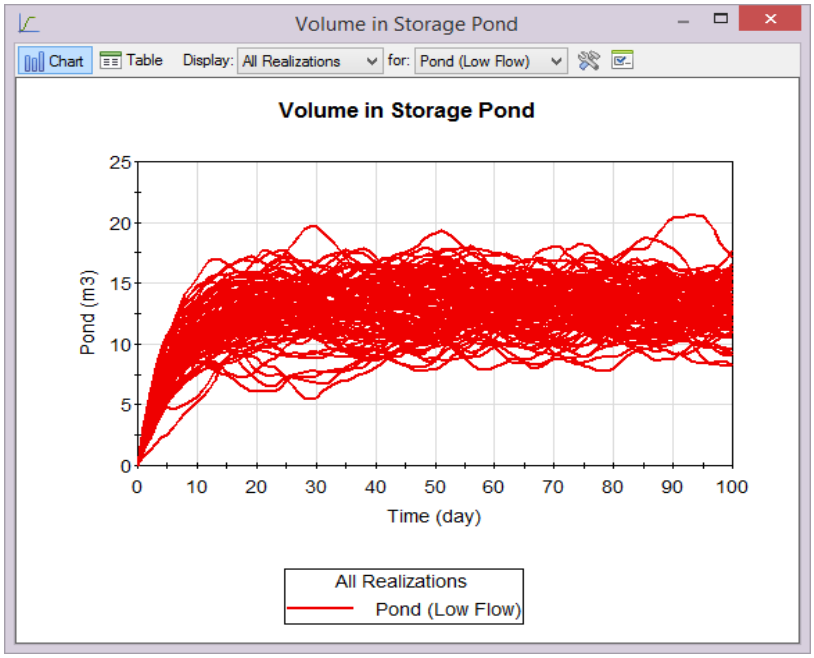


In particular, if **Realization**, **Statistic** or **Custom Statistics** is selected, GoldSim shows the selected result for each scenario:



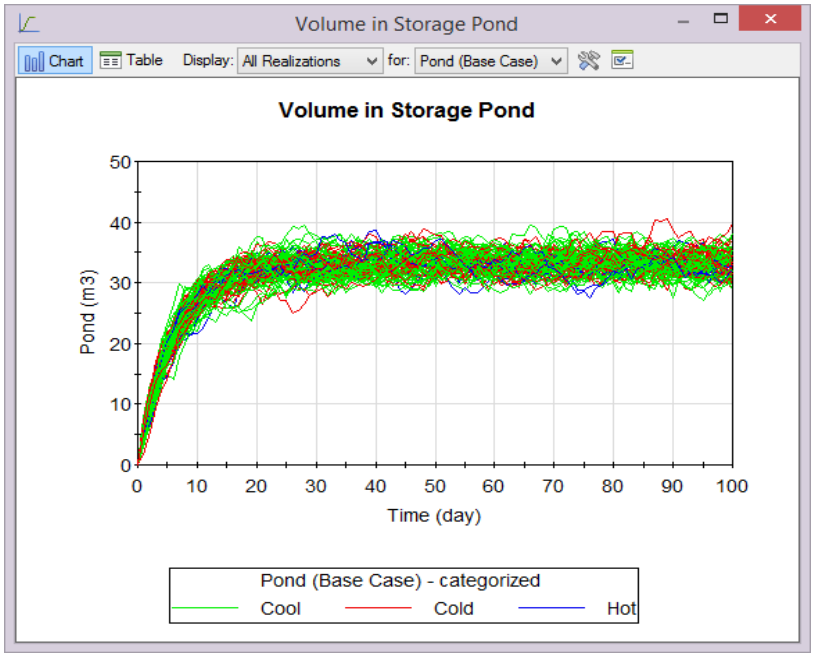
This is possible because in these three cases, there is only one set of results to display for each scenario (e.g., the selected statistic for each output).

However, if **All Realizations** or **Probabilities** is selected, only one scenario can be viewed at a time, and it is necessary to select which scenario (and output if multiple outputs are referenced by the Result element) you wish to view, since these two displays require multiple sets of results for each scenario (e.g., all realizations for a selected scenario and selected output):



The result and scenario that is displayed is selected directly to the right of the **Display** list (after "**for**"). In the example above, the result named "Pond" for scenario "Low Flow" has been selected.

When viewing a Time History result in Scenario Mode, if 1) you have defined more than one category (defined at the bottom of the Monte Carlo Result Display Properties dialog); and 2) you are displaying **All Realizations** for a multiple realization run, GoldSim will label the curves in the Time History Chart based on the categories you have defined. For example, if your categories were defined as Hot, Cool and Cold, a chart with categories might look like this:



Note that the chart is displaying a single result for a single scenario.

In addition, screening by category is active in Scenario Mode. That is, if you have chosen to ***screen*** out one or more categories (by clearing the **Include** box in the dialog above), the scenario results that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include.

Note that depending on how you have defined the categories, it is possible for a realization to fall into a different category in different scenarios. Hence, when screening results in Scenario Mode, the number of realizations within each scenario may differ. In the example below (displayed as a table), realization #18 is included in the first two scenarios, but is screened from the third:

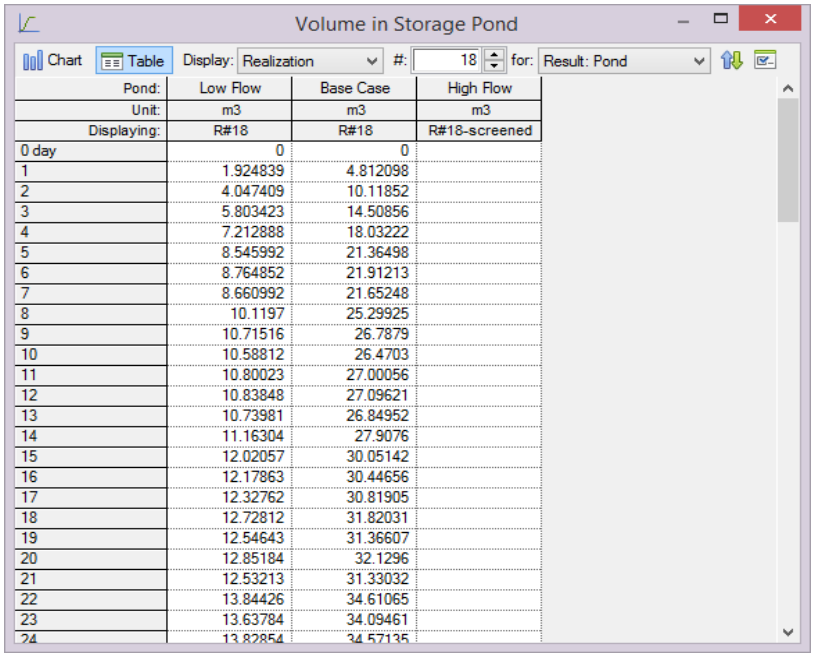

Volume in Storage Pond

Chart | Table | Display: Realization | #: 18 | for: Result: Pond

Pond:	Low Flow	Base Case	High Flow
Unit:	m3	m3	m3
Displaying:	R#18	R#18	R#18-screened
0 day	0	0	
1	1.924839	4.812098	
2	4.047409	10.11852	
3	5.803423	14.50856	
4	7.212888	18.03222	
5	8.545992	21.36498	
6	8.764852	21.91213	
7	8.660992	21.65248	
8	10.1197	25.29925	
9	10.71516	26.7879	
10	10.58812	26.4703	
11	10.80023	27.00056	
12	10.83848	27.09621	
13	10.73981	26.84952	
14	11.16304	27.9076	
15	12.02057	30.05142	
16	12.17863	30.44656	
17	12.32762	30.81905	
18	12.72812	31.82031	
19	12.54643	31.36607	
20	12.85184	32.1296	
21	12.53213	31.33032	
22	13.84426	34.61065	
23	13.63784	34.09461	
24	13.82854	34.57135	

Read more: Using Result Classification and Screening in Time History Results (page 571).

Note: High resolution results (unscheduled updates) are never displayed in Scenario Mode. They are only available in Result Mode. Hence, you cannot view high resolution results when comparing scenario results. When comparing scenarios, only scheduled updates are included in displays.

Read more: Viewing Unscheduled Updates in Time History Result Elements (page 586).

Note: Scenario results are not saved for time history results associated with a SubModel. If a Result element in a parent model is linked to SubModel time history results, no results will be displayed in Scenario Mode.

Read more: Viewing SubModel Results in Time History Result Elements (page 573).

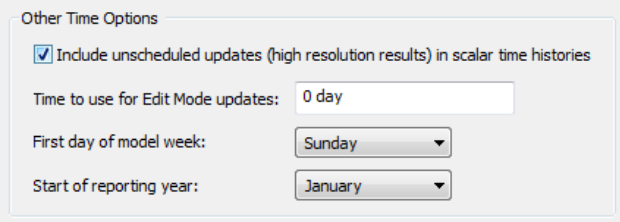
Viewing Unscheduled Updates in Time History Result Elements

In some cases, events or other changes in the model may not fall exactly on a scheduled update. That is, some events or changes may actually occur between scheduled updates of the model. These trigger an "unscheduled update" of the model. Unscheduled updates are timesteps that are dynamically inserted by GoldSim during the simulation in order to more accurately simulate the system.

Read more: Understanding Timestepping in GoldSim (page 415).

Unlike scheduled updates, unscheduled updates do not normally appear in time history plots and tables. That is, although these timesteps may affect the results (e.g., by making them more accurate at the scheduled timesteps), unscheduled updates of the model are not saved and displayed. Only the scheduled updates are actually saved and displayed.

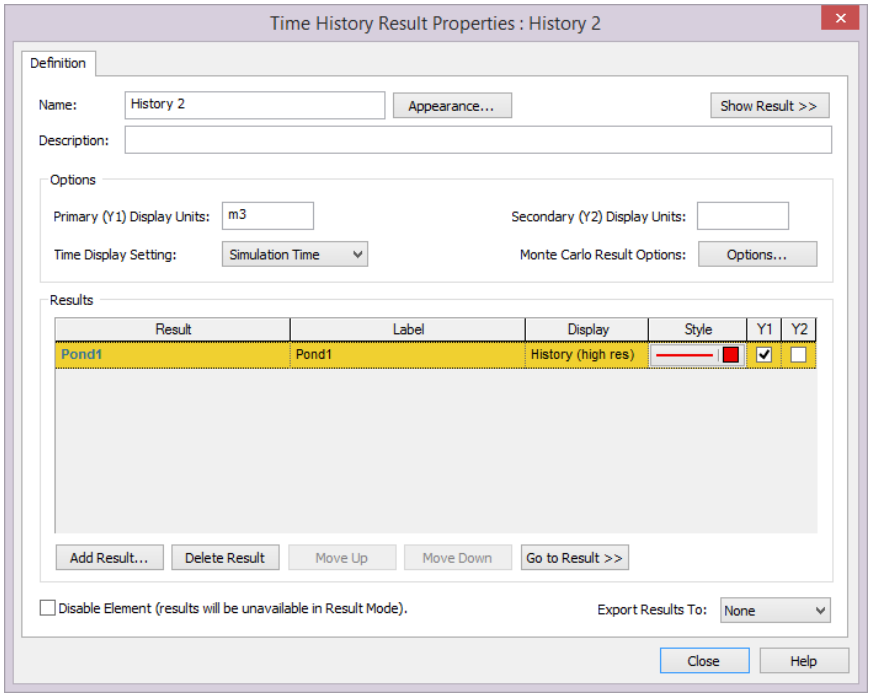
In some cases, however, it may be of interest to see the values of selected outputs at unscheduled updates. To facilitate this, GoldSim provides an option in the Advanced Time settings to save results at unscheduled updates (referred to as "high resolution" results). This option appears at the bottom of the Advanced Time Settings dialog (accessed from the **Advanced…** button on the **Time** tab of the Simulation Settings dialog):



By default, the checkbox labeled **Include unscheduled updates (high resolution results) in scalar time histories** is cleared. If you check this box, GoldSim will save unscheduled updates for time history results (referred to as "high resolution" results) under the following conditions:

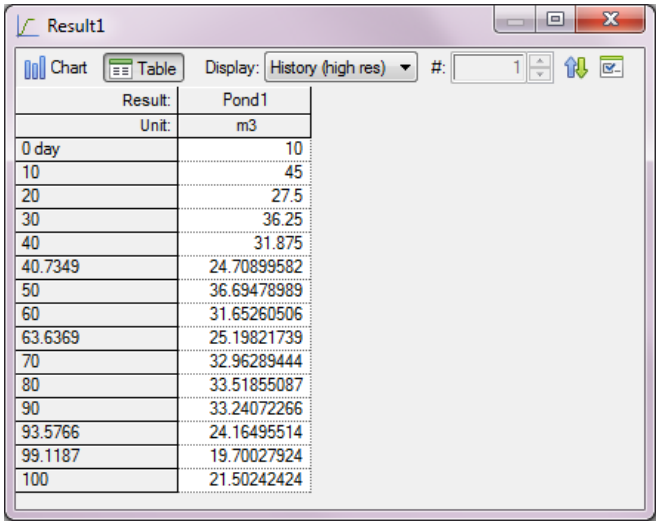
- High resolution results can only be viewed from within Time History Result elements. If the output is not referenced by a Time History Result element, high resolution results will not be displayed.
- High resolution results are only available for single realization runs (Deterministic simulations or probabilistic simulation with a single realization).
- High resolution results are only available for scalar outputs.
- The **Time Display Setting** in the Result Properties dialog for the Time History Result element must be set to "Simulation Time". High resolution results are not available when viewing Reporting Period-based results.
- High resolution results are only available for results that are added to the Result element before you run the model. If you add a result after you run the model, it will only show results at scheduled updates.

If these conditions are met, and you view a Time History Result element, the Result Properties will look like this:



If all necessary conditions are met, the **Display** column will list "History (high res)", indicating that the Result element will display high resolution results for that output.

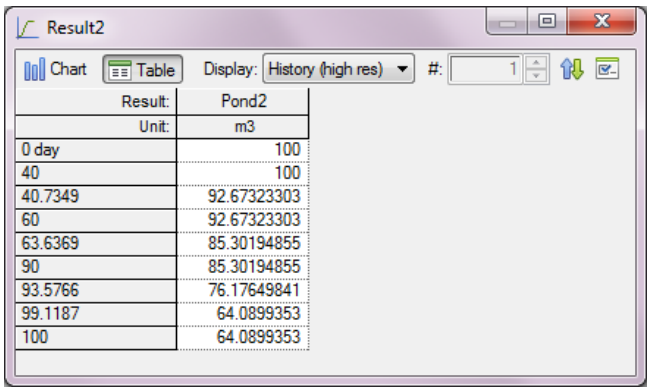
In such a case, the display will include unscheduled updates. In the example below (showing a table display), scheduled updates occur every 10 days (there is a Basic Step of 10 days), but unscheduled updates occur at 40.7 days, 63.6 days, 93.6 days and 99.1 days:



Result:	Pond1
Unit:	m3
0 day	10
10	45
20	27.5
30	36.25
40	31.875
40.7349	24.70899582
50	36.69478989
60	31.65260506
63.6369	25.19821739
70	32.96289444
80	33.51855087
90	33.24072266
93.5766	24.16495514
99.1187	19.70027924
100	21.50242424

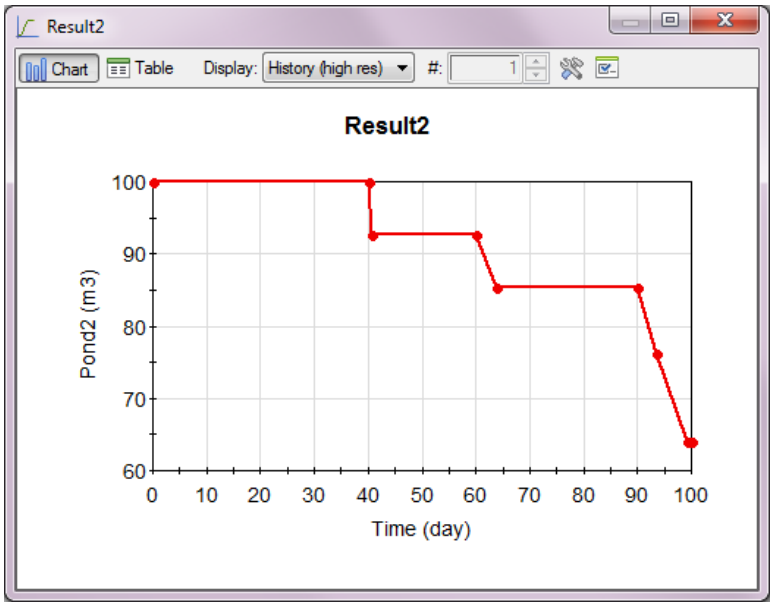
Note that when saving "high resolution" results, to reduce storage requirements, GoldSim does not actually save every unscheduled and scheduled update. Rather, the GoldSim only saves results 1) at each update for which the result is different from the previous update; and 2) the previous update (i.e., the update immediately before the update where the value changed). This information is sufficient to create an accurate chart display. All other updates (representing updates during which the value did not change from the previous value) do not need to be saved.

In the example above, the value is changing continuously. Hence, all scheduled and unscheduled updates must be saved. In the example below, however, the value only changes at unscheduled updates (and remains unchanged in between), and hence there is no reason to save all scheduled updates; only those immediately preceeding a change (at an unscheduled update) are saved:



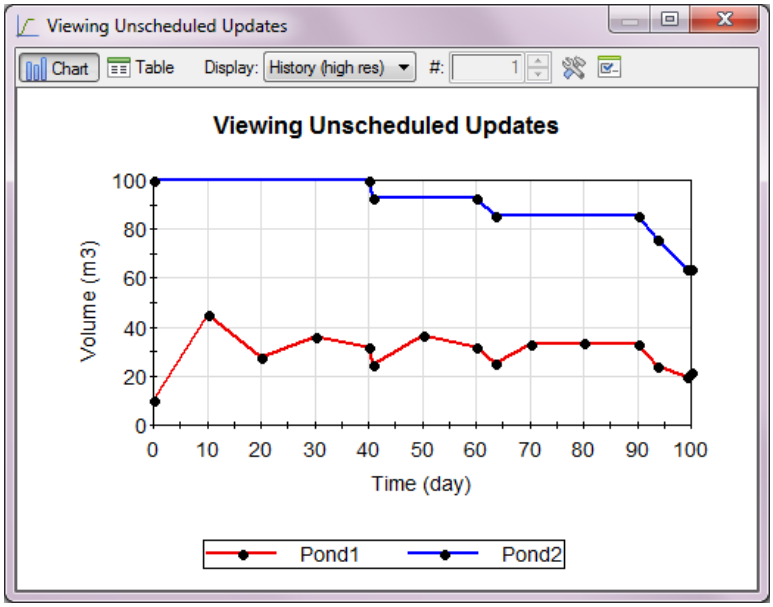
Result:	Pond2
Unit:	m3
0 day	100
40	100
40.7349	92.67323303
60	92.67323303
63.6369	85.30194855
90	85.30194855
93.5766	76.17649841
99.1187	64.0899353
100	64.0899353

The chart corresponding to the table shown above would look like this (symbols are placed at locations where a plot point occurs):



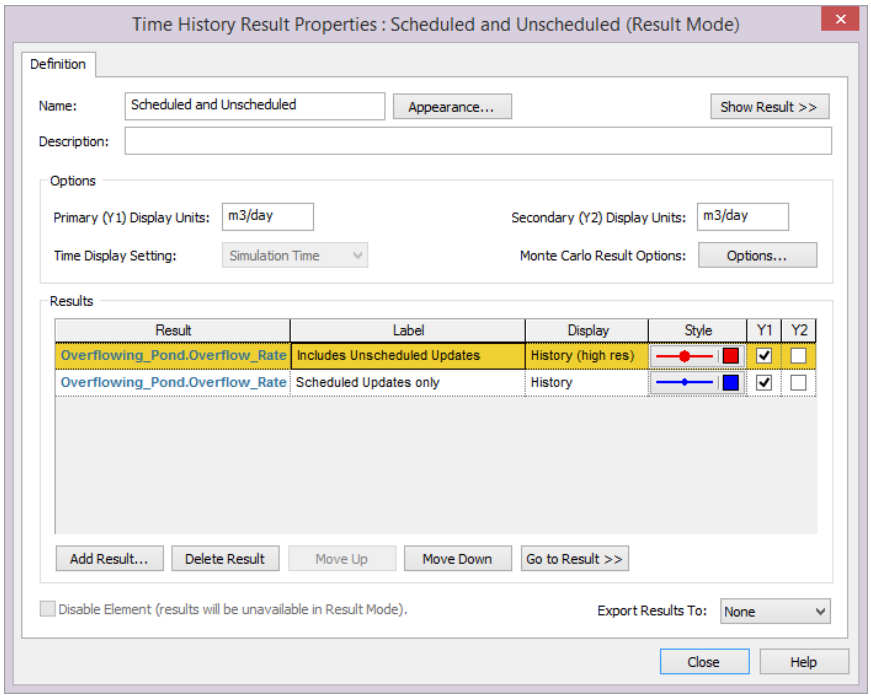
Note that there are no plot points for periods over which the plotted line is horizontal (since the value has not changed) even though there may have been scheduled (and unscheduled) updates during that period that did not affect the value.

The chart below displays the results for Pond1 and Pond2 (from the examples above) in the same chart:



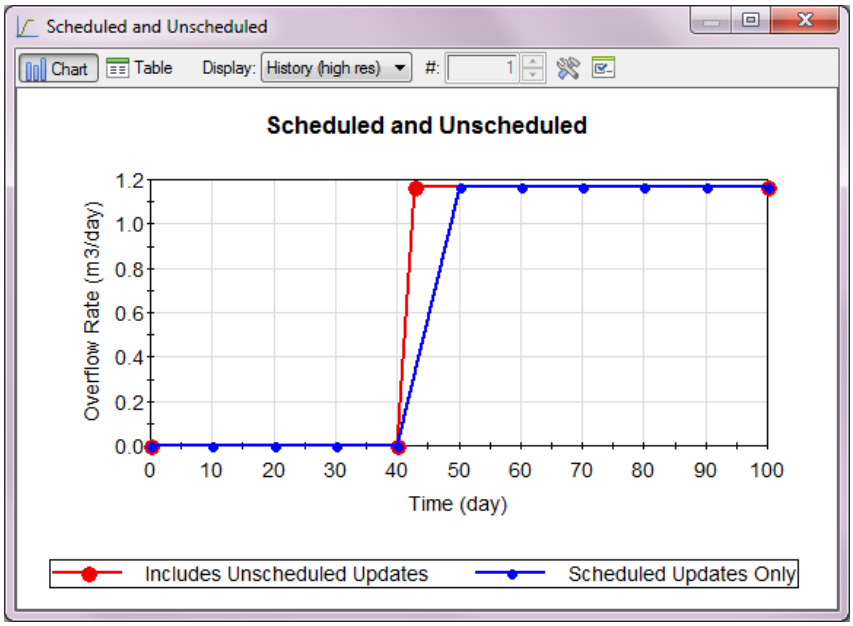
Note that Pond1 actually has more plot points than Pond2 (since it is changing continuously).

In some cases, it may be of value to view the same result with and without unscheduled updates. You can do this by taking advantage of the fact that high resolution results are only available for results that are added to the Result element *before* you run the model. If you add a result after you run the model, it will only show results at scheduled updates. Hence, to do such a comparison, you would 1)check **Include unscheduled updates (high resolution results) in scalar time histories** in the Advanced Time Settings dialog; 2) add a result to a Result element; 3) run the model; and 4) add the same result again after running the model. When you do this, the Result element will look like this:

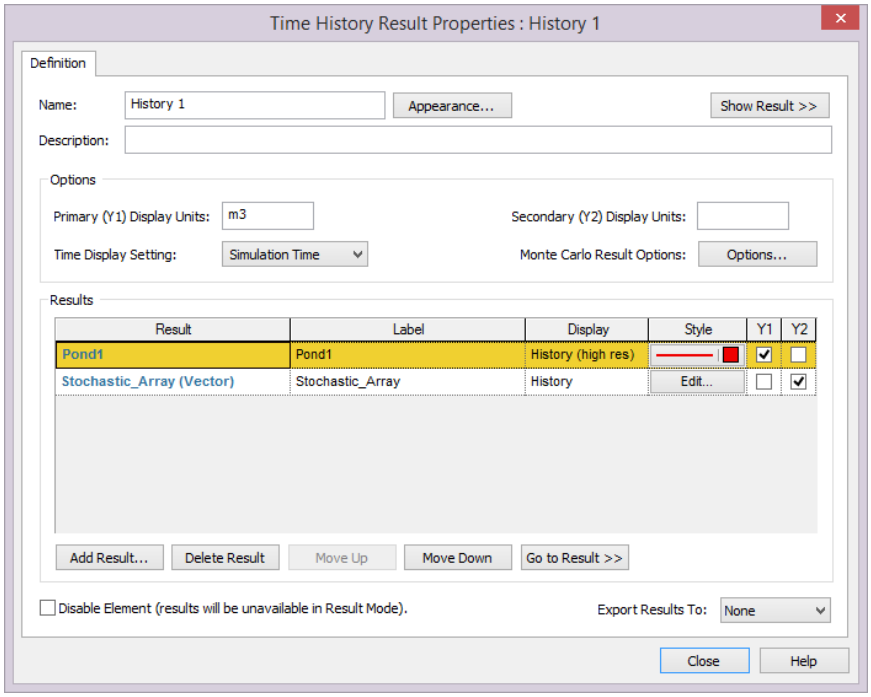


Noe that the first item (added before the simulation was run) will display high resolution results, while the second item (added after the simulation was run) will display a normal time history.

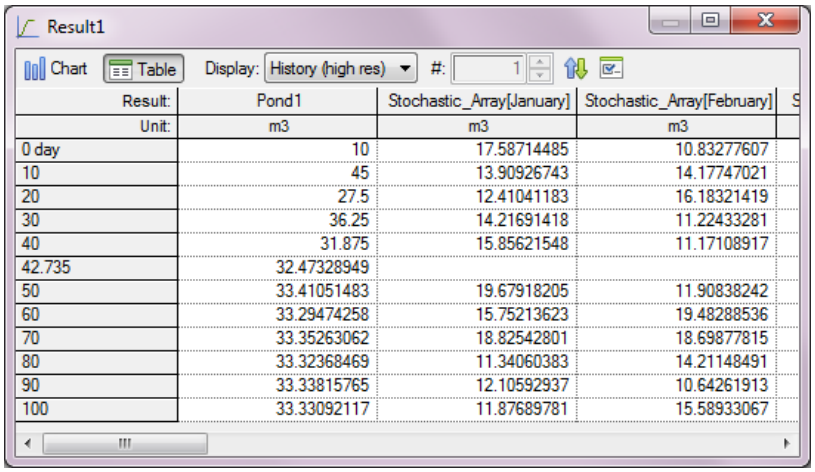
In this example, we have a 10 day timestep, and a Reservoir starts to overflow in the middle of a timestep (at about 43 days). This causes an unscheduled update to be inserted:



If your Time History Result element included one scalar output and one array, the scalar would provide high resolution results, while the array would not. This would be indicated in the Result Properties dialog:



The Table display would show blank cells for the array (at unscheduled updates):



Result:	Pond1	Stochastic_Array[January]	Stochastic_Array[February]
Unit:	m3	m3	m3
0 day	10	17.58714485	10.83277607
10	45	13.90926743	14.17747021
20	27.5	12.41041183	16.18321419
30	36.25	14.21691418	11.22433281
40	31.875	15.85621548	11.17108917
42.735	32.47328949		
50	33.41051483	19.67918205	11.90838242
60	33.29474258	15.75213623	19.48288536
70	33.35263062	18.82542801	18.69877815
80	33.32368469	11.34060383	14.21148491
90	33.33815765	12.10592937	10.64261913
100	33.33092117	11.87689781	15.58933067

Note that in this case, the high resolution results include all scheduled upates (since a row is provided for each scheduled update point due to the presence in the table of the array).

Note: High resolution results (unscheduled updates) are never displayed in Scenario Mode. They are only available in Result Mode. Hence, you cannot view high resolution results when comparing scenario results. When comparing scenarios, only scheduled updates are included in displays.

Read more: Viewing Scenario Results in Time History Result Elements (page 578).

A simple example file illustrating display of unscheduled updates (UnscheduledTimeSteps.gsm) can be found in the Timestepping subfolder of the the General Examples folder in your GoldSim directory.

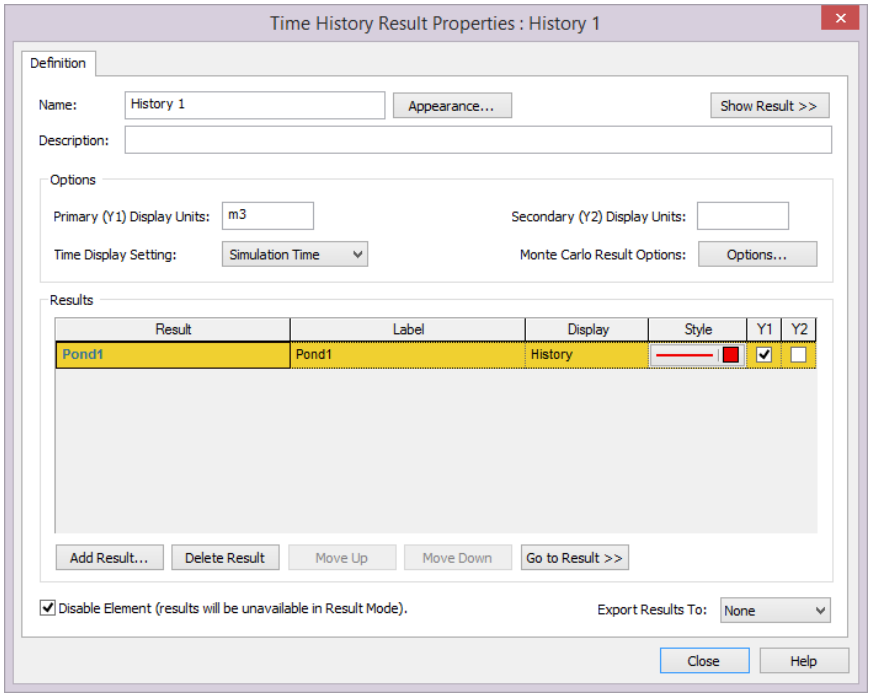
Disabling a Time History Result Element

If an output is linked to a Time History Result element, GoldSim automatically saves the results for the element (even if you have manually specified in the element's property dialog or the property dialog for a parent Container that time histories are not to be saved).

Read more: Saving Outputs as Results (page 453); Controlling Result Flags for Elements in the Container (page 145).

Under some circumstances, you may want to temporarily disable saving the results for an element that is linked to a Time History Result element (e.g., if you wish to run a large number of realizations).

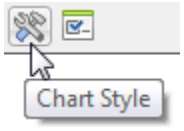
You can do this by checking the **Disable Element** checkbox at the bottom of the Time History Result element dialog:



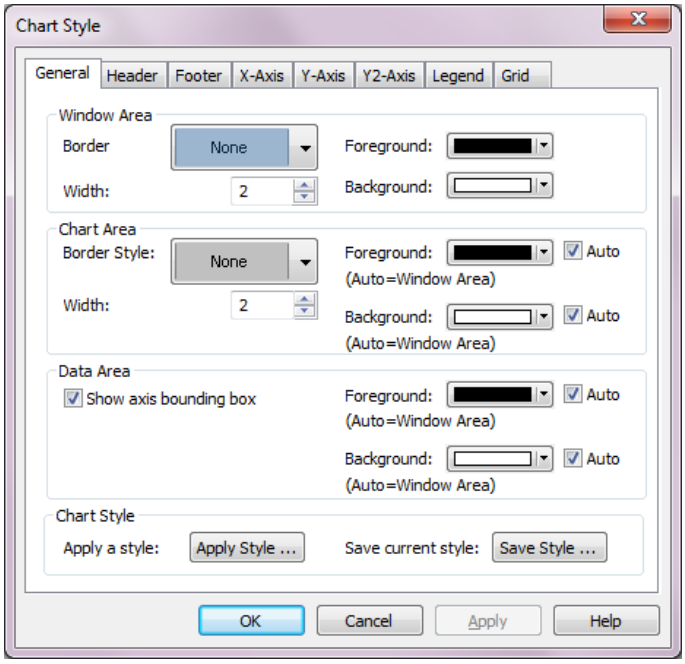
Note that you can enable or disable all of the Time History Result elements within a Container directly from the Container's property dialog.

Controlling the Chart Style in Time History Results

GoldSim has powerful charting capabilities that allow you to customize the appearance of each chart. This includes adding headers and footers, and changing axis scales and labels. Most of these attributes can be edited by pressing the **Edit Chart Style** button at the top of the Time History Chart window:



Pressing this button (or right-clicking in a chart and selecting **Edit Chart Style…**) provides access to the following dialog for editing the various chart properties:



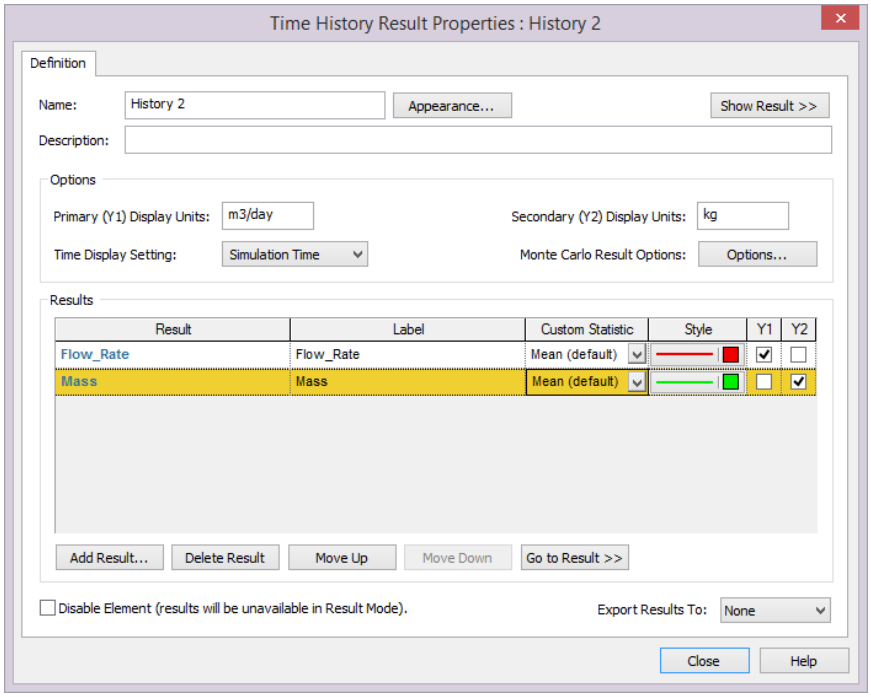
This dialog is common to all types of charts, and is discussed elsewhere.

Read more: Editing the Appearance of a Chart (page 661).

One key attribute for charts is how values on axes are displayed. You cannot control the number of significant figures displayed (this is automatically determined). You can, however, control under what circumstances scientific notation is used via the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).

Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 524).

In addition to the basic chart attributes controlled by the Chart Style dialog, the Time History Result Properties dialog itself is used to control some of the attributes of a chart:

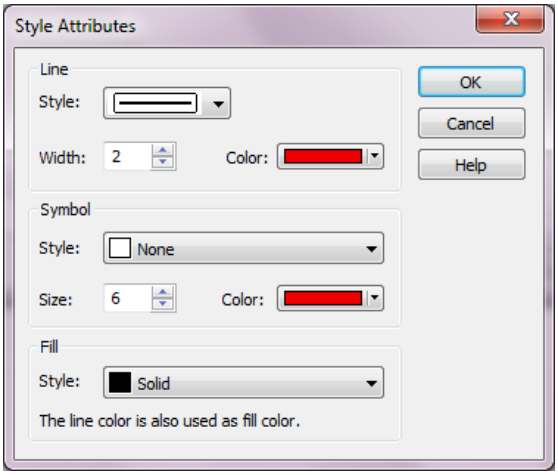


In particular,

- The units for each axis default to the Display Units of the first result added to each axis. However, you can change the **Display Units** that are displayed from within the Result Properties dialog.
- Only those results in which either the **Y1** or **Y2** box is checked will be included in displays. Hence, after adding a result to the list, you can temporarily hide it from displays by clearing one of these boxes.
- The **Label** is user-editable, and is used in legends (and column headers for tables).

Note: You can hide or show the legend on a chart via the context menu (i.e., accessed by right-clicking in the chart).

- The Style of each line displayed in a Time History Chart can be edited by clicking on the field (in the **Style** column) corresponding to each result. This provides access to the following dialog for editing the line style (for a distribution) or the fill style (for a single realization display):



Read more: Editing Data Styles (page 669).

If you are plotting an array, the Data Style for the different array items is defined in the Array Label Set dalog.

Read more: Viewing Time Histories for Array Outputs (page 547).

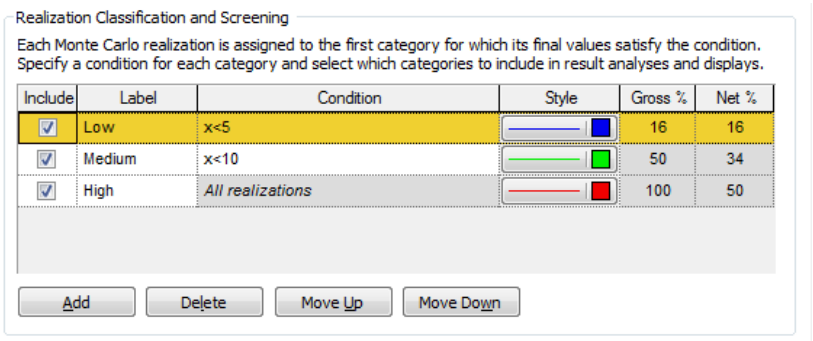
If you have run multiple realizations and you are displaying **Probabilities**, the Data Style for the display is specified at the top of the Monte Carlo Resilt Display Propertites dialog (in the section labled "History Statistics"). This dialog is accessed by pressing the **Options…** button in the Result Properties dialog of a Time History Result element.

Read more: Viewing Probability Histories for Multiple Realizations (page 555).

When viewing a Time History result, if 1) you have run multiple realizations; 2) you have defined more than one category; and 3) you are displaying **All Realizations**, GoldSim will label the curves in the Time History Chart based on the categories you have defined.

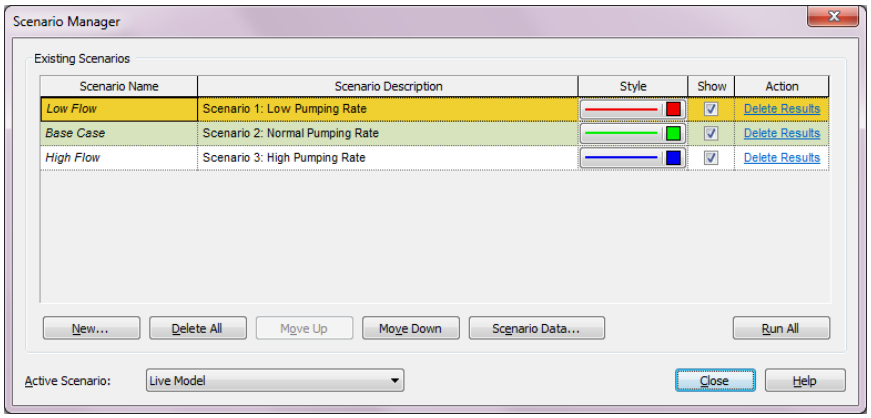
Read more: Using Result Classification and Screening in Time History Results (page 571).

The chart will display all realizations for a single result, with each curve having a style defined by its category. For each category, you can edit the **Style** via the Monte Carlo Result Display Properties dialog (accessed by pressing the **Options…** button in the Result Properties dialog of a Time History Result element):



The Style used for each category displayed in a Time History Chart can be edited by clicking on the field (in the **Style** column) corresponding to each category.

When a model is in Scenario Mode, Time History Result elements can be used to view scenario results. In this case, the line style for time history results for the different scenarios is controlled by the Scenario Manager dialog (accessed from the main menu (**Run | Scenario Manager…**) or by pressing **F7**):



Read more: Viewing Scenario Results in Time History Result Elements (page 578).

In order for scenario results to be displayed in a Time History Result element, the **Show** button must be checked for each scenario to be displayed in the Scenario Manager.

Note that for each scenario, you can edit the **Style**, as well as the **Scenario Name**, which is used used in legends (and column headers for tables).

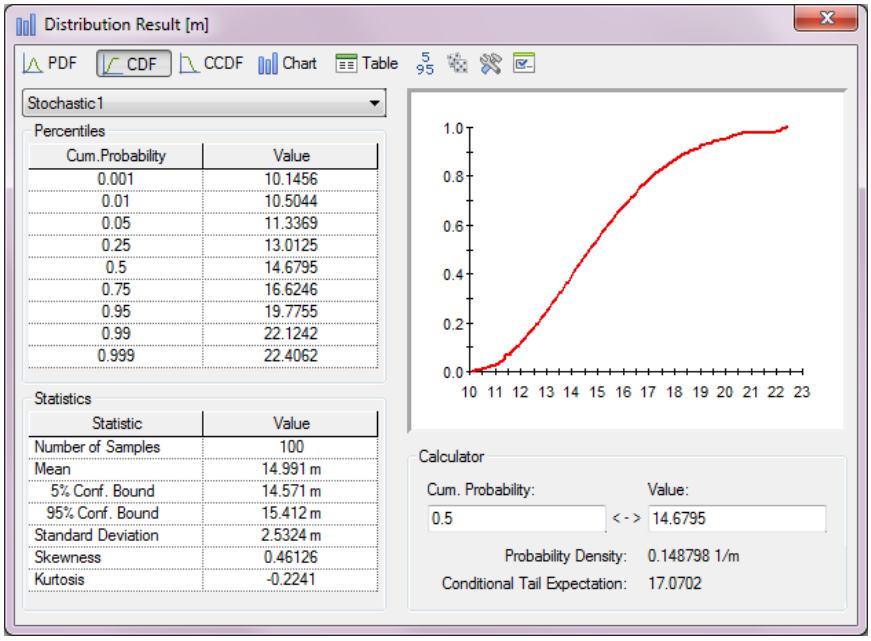
Viewing Distribution Results

Distribution results provide a way to view the final values of probabilistic outputs. Distribution results can only be viewed for scalar outputs (or scalar items of arrays).

A Distribution result has three types of views:

- Distribution Summary;
- Chart View; and
- Table View.

If you have saved Final Values for an output, you can display a Distribution result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Distribution Result…** from the context menu. By default, a Distribution Summary will be displayed:

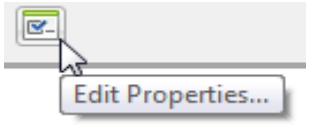


Note: When viewing a Distribution Result element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

Read more: Creating and Using Result Elements (page 526).

Viewing the Properties of a Distribution Result

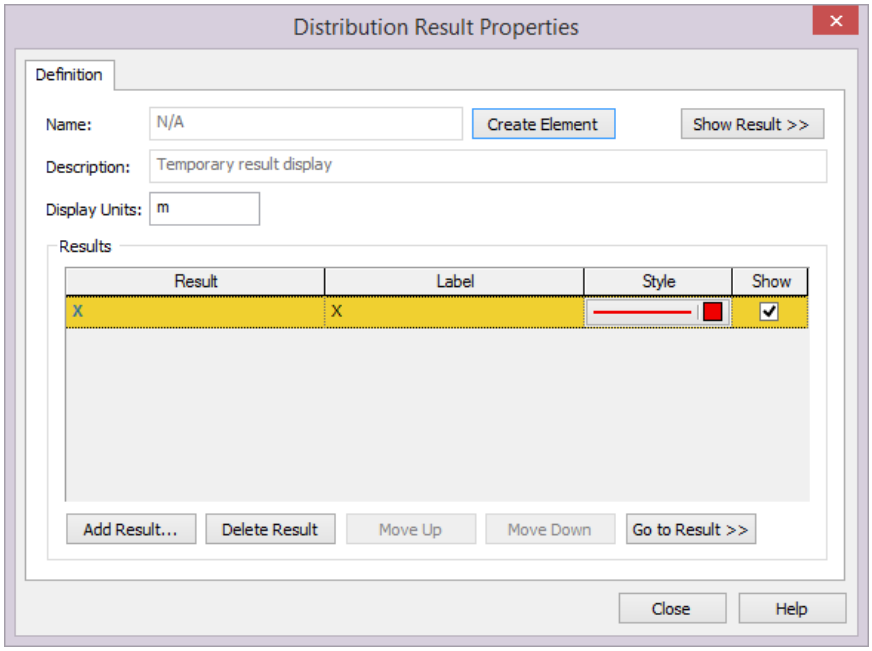
If you click on the Result Properties button when viewing a Distribution Result chart or table, the Properties dialog for the result will be displayed. The Result Properties button is the furthest button to the right at the top of the display window:



Note that when you press this button, it does not close the result display window you were displaying. Both windows are displayed simultaneously (although you can subsequently close one or the other). When viewing results, any changes you make to the Result Properties are immediately represented in the result display.

The Result Properties dialog is always modal. That is, it cannot be minimized, and with the exception of viewing the result display (which can be displayed simultaneously and share the focus with the Properties dialog), it retains the focus while it is displayed. As a result, you must close the Result Properties dialog before you can edit any other part of your GoldSim model.

The Result Properties dialog for an interactive Distribution result looks like this:



At the top of the page is a button to **Create Element**. By pressing this button, a Result element is created. When you do so, the key change that will be noticeable is that the **Name** will now be editable. The **Create Element** button becomes an **Appearance** button (which is used to modify the appearance of the element itself).

Read more: Creating and Using Result Elements (page 526).

The **Show Result >>** button opens a result display window (while keeping the Properties dialog simultaneously open.

Read more: Viewing a Distribution Summary (page 600); Viewing a Distribution Chart (page 604); Viewing a Distribution Table (page 608).

The **Add Result…** button allows you to add other results to the chart. Results can be deleted with the **Delete Result** button. In addition to adding standard outputs (such as Expressions or the primary output of a Reservoir element), you can also add Distribution outputs, which are complex outputs that represent all the statistical information necessary to define a probability distribution (and can ony be produced by certain elements).

Read more: Adding a Distribution Output to a Distribution Result (page 620).

The order in which multiple results appear in the list can be changed using the **Move Up** and **Move Down** buttons.

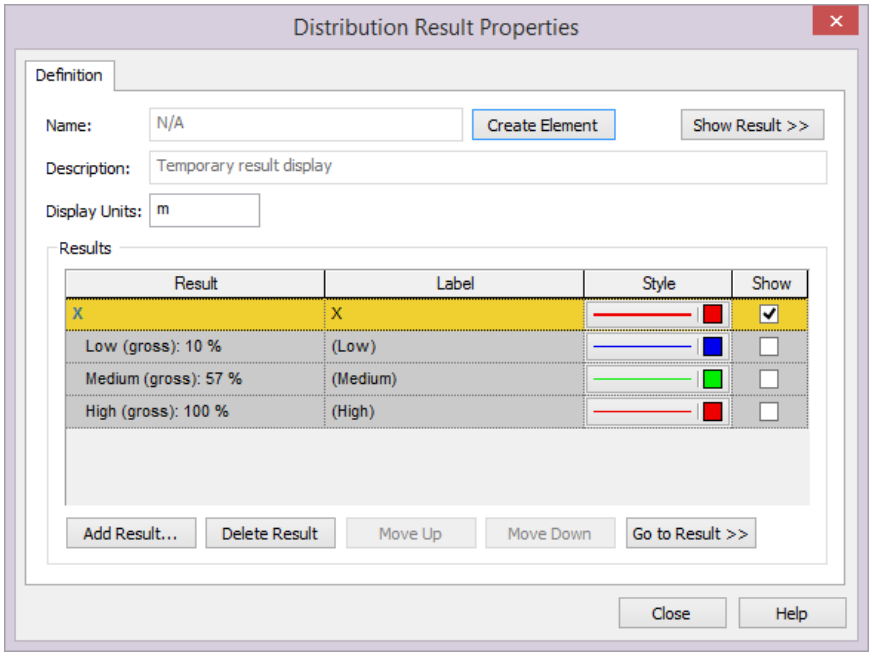
Read more: Viewing Distributions of Multiple Outputs (page 611).

For each result in the list, you specify a **Style** (used in charts), whether or not the result is to be included when displaying results (**Show**), and the **Label** used in legends in charts and headers in tables.

Read more: Controlling the Chart Style in Distribution Results (page 626).

You can also specify the **Display Units** for the X-axis of the result (which overrides the display units specified within the element's property dialog).

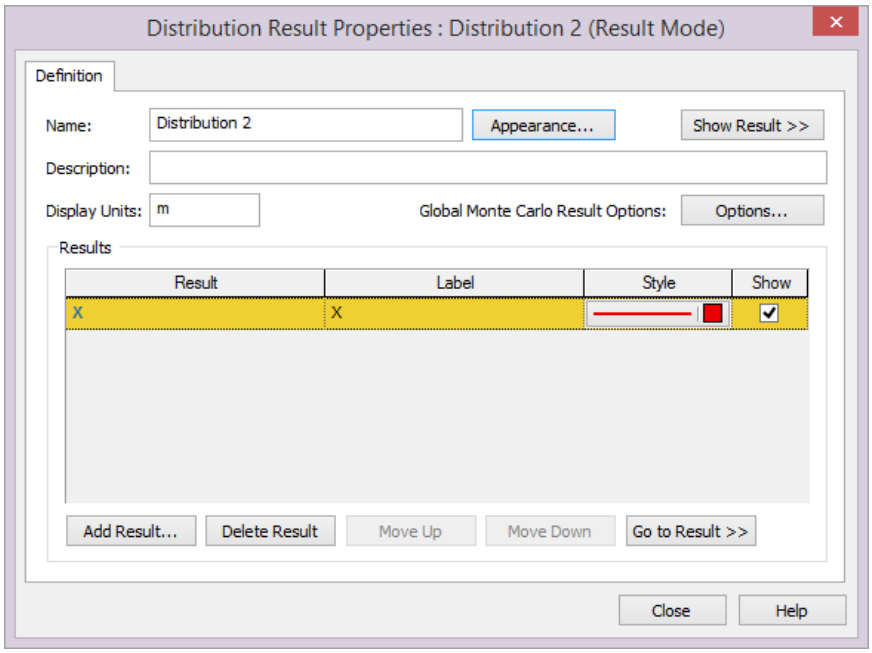
If you have created categories (and have specified only a single result in the list), you will note that the various categories will also be listed in the dialog:



Categories allow you to classify the realizations into various groups in order to analyze the results.

Read more: Using Result Classification and Screening in Distribution Results (page 616).

The Properties dialog for a Distribution Result element has several differences:



- The **Create Element** button is replaced by an **Appearance** button (which is used to modify the appearance of the element itself).
- The **Name** (and **Description**) are editable (since this is an element).
- An **Options…** button is available that provides access to the Monte Carlo Result Display Properties.

Read more: Controlling Monte Carlo Result Options (page 441).

Viewing a Distribution Summary

If you have saved Final Values for an output, you can display a Distribution Result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Distribution Result…** from the context menu. By default, a Distribution Summary will be displayed.

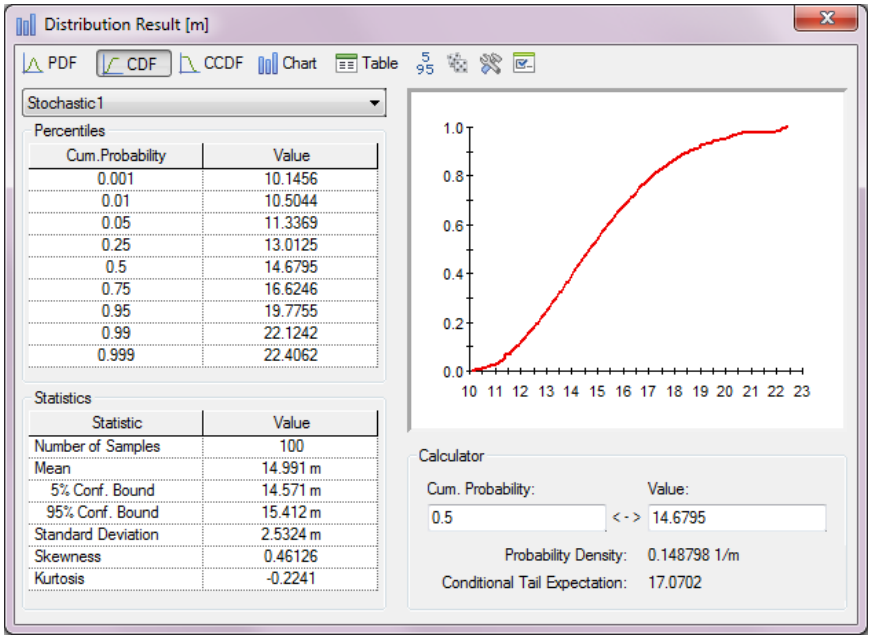
If you are viewing a different type of display (either a chart or table), you can view a Distribution Summmary by pressing the **Chart** or **Table** button at the top of the display (i.e., making sure both are buttons are cleared).

Note: When viewing a Distribution Result element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

The Distribution Summary view of a distribution result is similar in appearance to the dialog used for defining a Stochastic element.

Read more: Specifying the Distribution for a Stochastic Element (page 158).

A Distribution Summary looks like this:

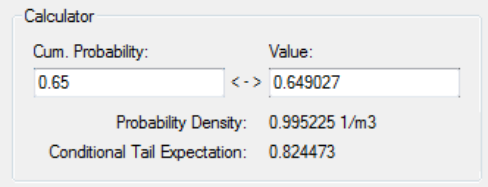


The *preview pane* (in the upper right-hand corner of the screen) shows a preview of the chart view of the distribution. You can copy the preview pane to the clipboard using **Ctrl+C**.

The left side of the screen displays a common set of percentiles for the distribution (in terms of Cumulative Probabily/Value pairs).

The statistics for the distribution (***Mean***, 5% and 95% Confidence Bounds on the Mean, ***Standard Deviation***, ***Skewness***, and ***Kurtosis***) are shown directly below the distribution's percentiles. The significance of these statistics is described in Appendix A. The number of realizations (**Number of Samples**) is also indicated.

The Calculator section of the window allows you to compute the value associated with a particular percentile or the percentile associated with a particular value:

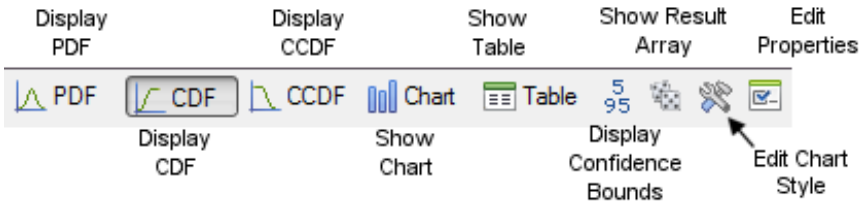


Enter the percentile in the Cum. Probability field and the corresponding value will be displayed in Value OR Enter the value in the Value field and the corresponding percentile will be displayed in Cum. Probability.

The Calculator also displays the Probability Density and the ***Conditional Tail Expectation*** for the specified Cumulative Probability/Value pair. The Conditional Tail Expectation is the expected value of the output given that it lies above a specified Cumulative Probability. That is, it represents the mean of the worst 100(1 - α)% of outcomes, where α is the specified Cumulative Probability.

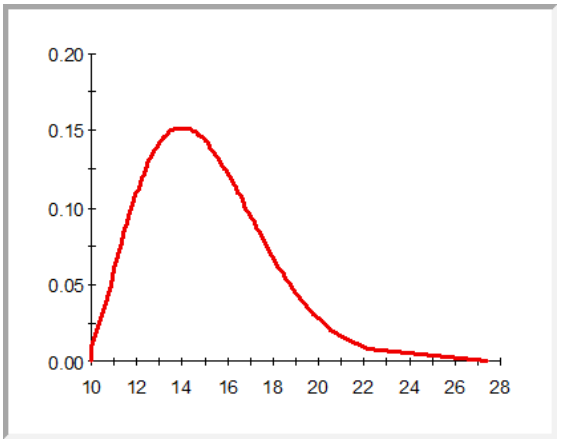
Note: Calculation of the Conditional Tail Expectation is discussed in detail in Appendix B.

The Distribution Summary has a variety of buttons at the top of the window, some of which are common to all result windows, and some of which are specific to Distribution Result windows:



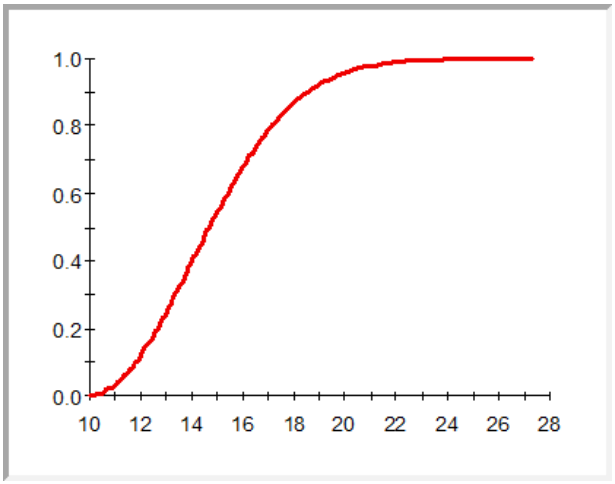
The functions of these buttons are as follows (only the buttons that are active in Distribution Summary view are described):

Display PDF: This displays the preview pane as a probability density function (PDF):

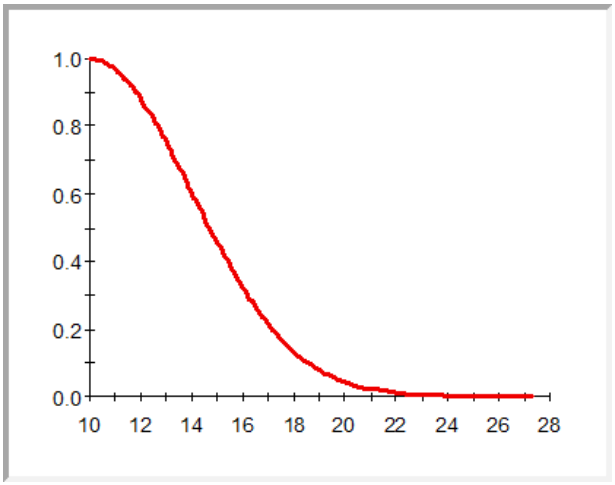


Note: The algorthim used to create a PDF plot is discussed in detail in Appendix B.

Display CDF: This displays the preview pane as a cumulative distribution function (CDF):



Display CCDF: This displays the preview pane as a complementary cumulative distribution function (CCDF):



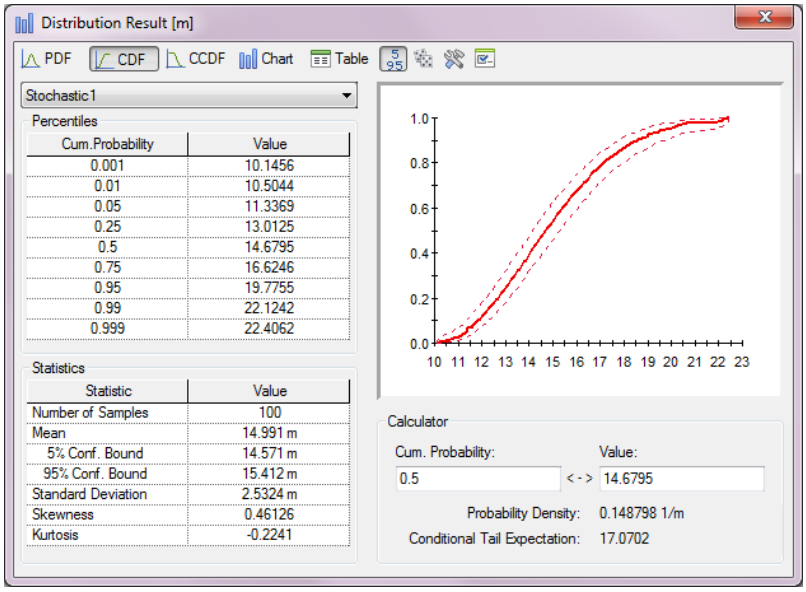
Show Chart: Selecting this button switches to a Chart view of the result. You can also toggle between the Distribution Summary View and the Chart View by double-clicking in the preview pane. Note that when viewing a Distribution Summary, both the Display Chart and the Display Table buttons appear deselected.

Read more: Viewing a Distribution Chart (page 604).

Show Table: Selecting this button switches to a Table view of the result. Note that when viewing a Distribution Summary, both the Display Chart and the Display Table buttons appear deselected.

Read more: Viewing a Distribution Table (page 608).

Show Confidence Bounds: If you press this button or press Ctrl+Shft+B, GoldSim will display confidence bounds on CDFs and CCDFs (they cannot be shown for PDFs):



These bounds reflect uncertainty in the probability distribution due to the finite number of Monte Carlo realizations (as the number of realizations is increased, the uncertainty in the distribution decreases). The confidence bounds represent the 5% and 95% confidence limits on the distribution. The calculation of confidence bounds is discussed in detail in Appendix B. Note that confidence bounds cannot be displayed if importance sampling has been used in the model.

Show Result Array: Pressing this button displays a new (modal) window containing the result array.The result array represents a sorted table of all of the results, and is used to produce the charts and statistics for the distribution. The table shows the value, the weight, the number of occurrences of the value, the cumulative probability of the value, and the 5% and 95% confidence bounds for the value at that cumulative probability. (Note, however, that confidence bounds cannot be computed if importance sampling has been used in the model.)

Read more: Viewing the Distribution Result Array (page 610).

Edit Properties: This provides access to the Result Properties.

Read more: Viewing the Properties of a Distribution Result (page 597).

Note: If you press the Copy button (or **Ctrl+C**) from the Summary View of a Distribution result, the preview pane chart is copied to the clipboard.

By default, Distribution results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display results at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the results at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 525).

Note: When viewing distribution results produced by a nested Monte Carlo simulation (using a SubModel), the Distribution Summary dialog is modified somewhat and provides slightly different options.

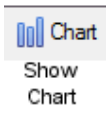
Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933).

Viewing a Distribution Chart

When you display a Distribution Result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Distribution Result…** from the context menu., by default, a Distribution Summary will be displayed.

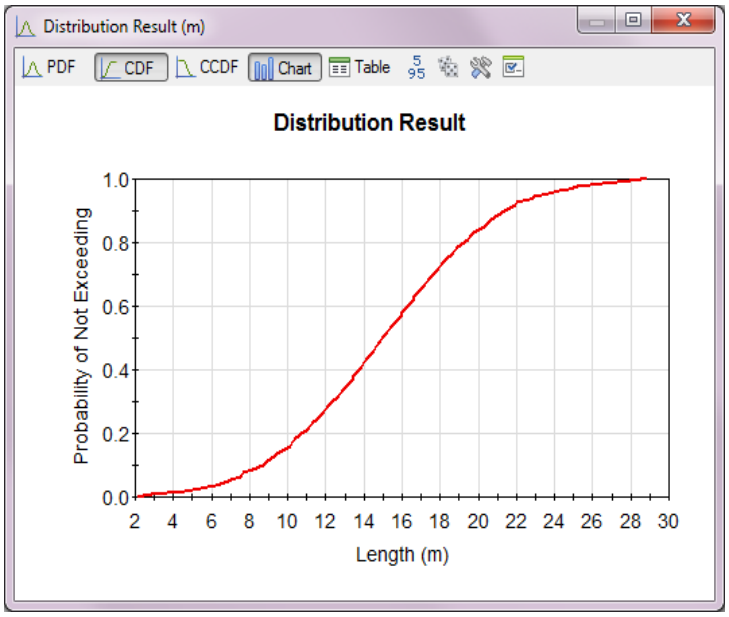
Read more: Viewing a Distribution Summary (page 600).

You can view a Distribution Chart by pressing the **Show Chart** button when viewing a Distribution Summary or a Distribution Table. You can also view a Distribution Chart by double-clicking on the preview pane in the Distribution Summary window.

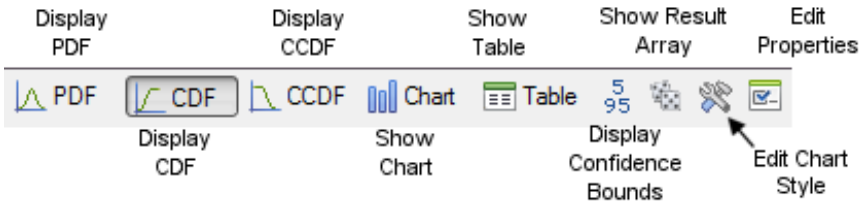


Note: When viewing a Distribution Result element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

A Distribution Chart looks like this:



The Distribution Chart has a variety of buttons at the top of the window, some of which are common to all result windows, and some of which are specific to Distribution Result windows:



The functions of these buttons are:

Display PDF: This displays the chart as a probability density function (PDF).

Note: The algorthim used to create a PDF plot is discussed in detail in Appendix B.

Display CDF: This displays the chart as a cumulative distribution function (CDF).

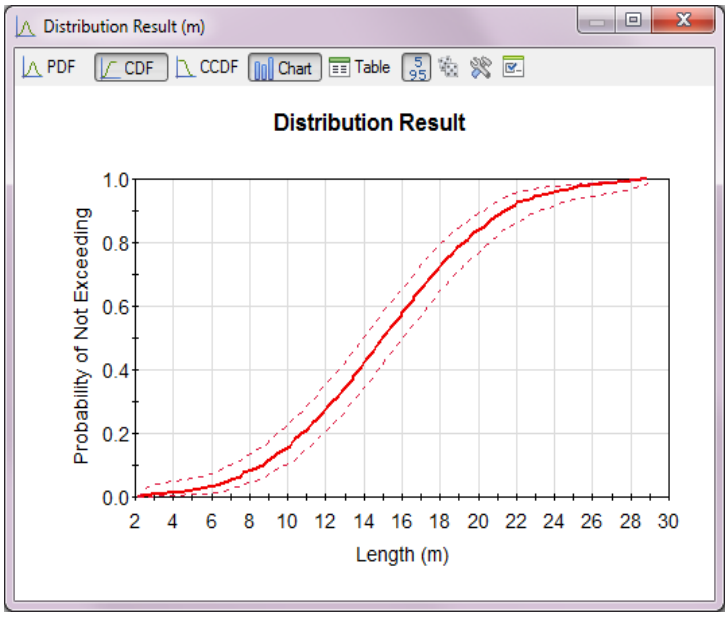
Display CCDF: This displays the chart as a complementary cumulative distribution function (CCDF).

Show Chart: Deselecting this button switches to a Distribution Summary view of the result. You can also toggle from the Chart View to the Distribution Summary View by double-clicking anywhere in the chart. Note that when viewing a Distribution Chart, the Display Chart button appears selected.

Show Table: Selecting this button switches to a Distribution Table view of the result. Note that when viewing a Distribution Table, the Display Table button appears selected.

Read more: Viewing a Distribution Table (page 608).

Show Confidence Bounds: If you press this button or press Ctrl+Shft+B, GoldSim will display confidence bounds on CDFs and CCDFs (they cannot be displayed for PDFs):



These bounds reflect uncertainty in the probability distribution due to the finite number of Monte Carlo realizations (as the number of realizations is increased, the uncertainty in the distribution decreases). The confidence bounds represent the 5% and 95% confidence limits on the distribution. The calculation of confidence bounds is discussed in detail in Appendix B. Note that confidence bounds cannot be displayed if importance sampling has been used in the model.

Show Result Array: Pressing this button displays a new (modal) window containing the result array.The result array represents a sorted table of all of the results, and is used to produce the charts and statistics for the distribution. The table shows the value, the weight, the number of occurrences of the value, the cumulative probability of the value, and the 5% and 95% confidence bounds for the value at that cumulative probability. (Note, however, that confidence bounds cannot be computed if importance sampling has been used in the model.)

Read more: Viewing the Distribution Result Array (page 610).

Edit Chart Style: This button provides access to a dialog for editing the chart style.

Read more: Controlling the Chart Style in Distribution Results (page 626).

Edit Properties: This provides access to the Result Properties.

Read more: Viewing the Properties of a Distribution Result (page 597).

Like all result charts, you can also control various attributes of the chart via a context menu. This includes the ability to turn on and off a legend. The legend uses the **Label** for the result defined in the Result Properties page. When viewing multiple results on one chart, you will always want to display the legend.

Read more: Using Context Menus in Charts (page 522); Viewing Distributions of Multiple Outputs (page 611).

When the results being plotted are conditions (i.e., True or False) or respresent discrete (as opposed to continuous) distributions, GoldSim uses special algorithms to plot these in a consistent and effective manner.

Read more: Plotting Condition Distributions (page 607); Plotting Discrete Distributions (page 607).

You can copy the chart to the clipboard by right-clicking in the chart, and selecting **Copy** from the context menu. Alternatively, you can press **Ctrl+C** while viewing the chart. To export a chart, right-click in the chart, and select **Export…** from the context menu.

Read more: Copying a Chart or Table (page 691); Exporting a Chart (page 691).

By default, Distribution results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display results at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the results at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 525).

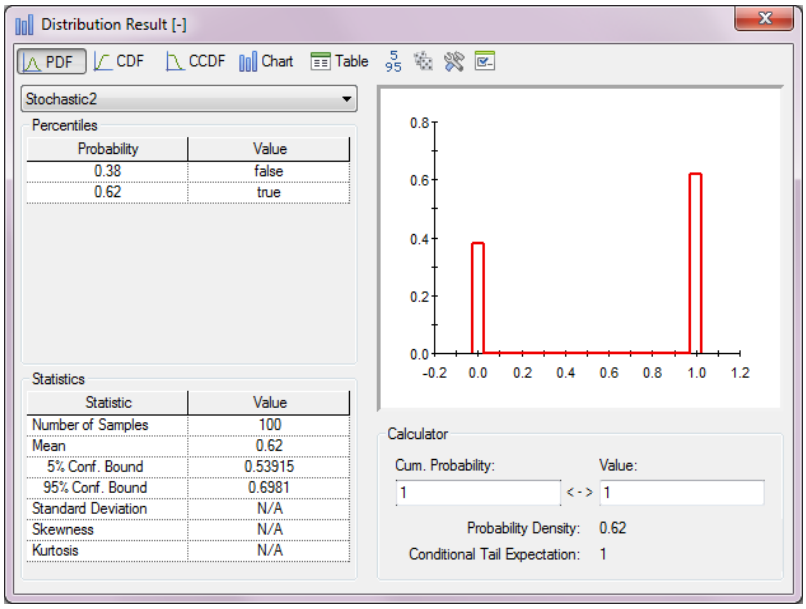
Note: When viewing distribution results produced by a nested Monte Carlo simulation (using a SubModel), the Distribution Chart display is modified somewhat and provides slightly different options.

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933).

Plotting Condition Distributions

Condition outputs are either True or False and are typically used as state variables or flags in a simulation. As a result, in some situations, you may want to save and plot distributions of conditions. To facilitate this, when plotting a distribution of a condition, GoldSim plots True as 1 and False as 0.

When viewing a distribution of conditions in the Distribution Summary window, within the Percentile portion of the window, GoldSim does not display percentiles; rather it displays the probability of True and False:



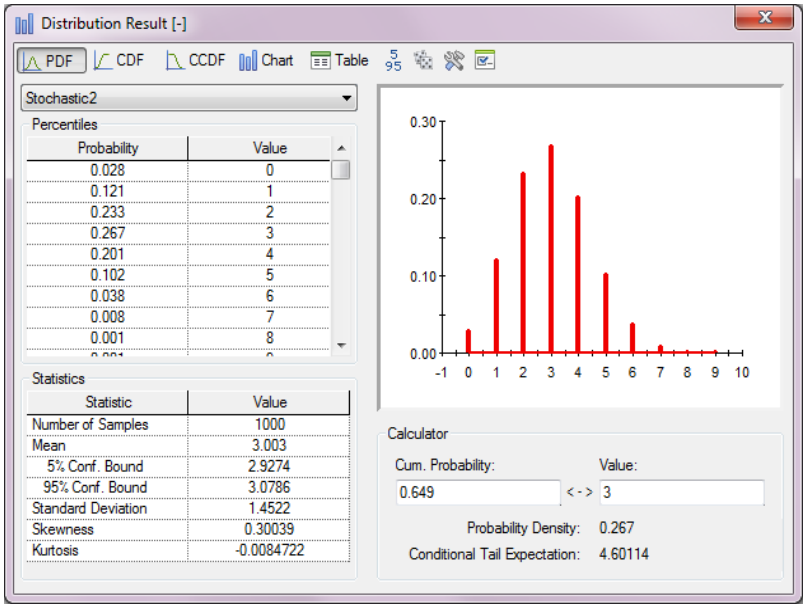
The displayed Mean is actually computed as the probability of being True. The other statistics are not computed. GoldSim also displays the 5% and 95% confidence bounds on the Mean.

Read more: Understanding Output Attributes (page 93); Viewing a Distribution Summary (page 600).

Plotting Discrete Distributions

Discrete distributions only take on a finite number of discrete values (i.e., they are not continuous). GoldSim's Monte Carlo engine includes algorithms that attempt to identify discrete (as opposed to continuous) results, and plot them as such.

When viewing a discrete distribution in the Distribution Summary window, within the Percentile portion of the window GoldSim does not display percentiles; rather, it displays the actual probability of each result:



Read more: Viewing a Distribution Summary (page 600).

Viewing a Distribution Table

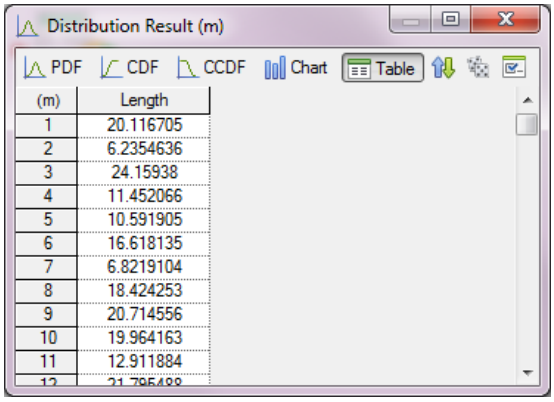
When you display a distribution result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Distribution Result…** from the context menu., by default, a Distribution Summary will be displayed.

Read more: Viewing a Distribution Summary (page 600).

You can view a Distribution Table by pressing the **Show Table** button when viewing a Distribution Summary or a Distribution Chart.

Note: When viewing a Distribution Result element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

A Distribution Table looks like this:



Each row of the table is a separate realization. The header row for a table shows the units for the result above the first column and the Label for the result. The Label is specified in the Result Properties dialog and defaults to the result (output) name.

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively. If the column is too narrow to display the contents of a cell, holding the cursor over the cell displays a tool-tip with the contents.

Tables have the following buttons (from left to right in the figure above):

Display PDF: Switches to Distribution Summary view and displays the preview pane as a probability density function (PDF).

Display CDF: Switches to Distribution Summary view and displays the preview pane as a cumulative distribution function (CDF).

Display CCDF: Switches to Distribution Summary view and displays the preview pane as a complementary cumulative distribution function (CCDF).

Show Chart: Selecting this button switches to a Distribution Chart view of the result.

Read more: Viewing a Distribution Chart (page 604).

Show Table: Deselecting this button switches to a Distribution Summary view of the result. Note that when viewing a Distribution Table, the Display Table button appears selected.

Sort: This button allows you to sort the rows based on a selected column.

Read more: Sorting Values in Result Tables (page 524).

Show Result Array: Pressing this button displays a new (modal) window containing the result array.The result array represents a sorted table of all of the results, and is used to produce the charts and statistics for the distribution. The table shows the value, the weight, the number of occurrences of the value, the cumulative probability of the value, and the 5% and 95% confidence bounds for the value at that cumulative probability. (Note, however, that confidence bounds cannot be computed if importance sampling has been used in the model.)

Read more: Viewing the Distribution Result Array (page 610).

Edit Properties: This provides access to the Result Properties.

Read more: Viewing the Properties of a Distribution Result (page 597).

Several additional points should be noted regarding Distribution Tables:

- Conditions are displayed in tables as either 1/0, true/false, True/False, TRUE/FALSE, on/off, or High/Low. You select which of these pairs to use on the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).
- You can control the number of significant figures displayed in tables from the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).

  Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 524).
- You can copy the contents of a distribution table to the clipboard by pressing **Ctrl+C**. You must first select the cells you wish to copy. You

can do so by placing your cursor in one cell and dragging to another location. You can select the entire table by pressing the cell in the upper lefet-hand corner of the table. Cells that are not adjacent can be selected using **Ctrl** key when selecting. Selected items will be highlighted in black. You can subsequently paste the table into another application (such as a spreadsheet).

Read more: Selecting Items and Copying Values in Result Tables (page 523).

By default, Distribution results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display results at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the results at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 525).

Viewing the Distribution Result Array

GoldSim allows you to view an alternative kind of table view of a distribution result referred to as the **result array**. This table is accessed by pressing the **Show Result Array** button in any of the Distribution Result display windows:

A typical result array is shown below:

Result Array: Stochastic1

#	Value	Weight	Count	Cum	Q05	Q95
1	10.15332	0.001	1	0.0005	10.13708	10.23274
2	10.18872	0.001	1	0.0015	10.13959	10.33935
3	10.27180	0.001	1	0.0025	10.14644	10.42781
4	10.32809	0.001	1	0.0035	10.16003	10.46948
5	10.36224	0.001	1	0.0045	10.18109	10.52375
6	10.43698	0.001	1	0.0055	10.22075	10.55351
7	10.44847	0.001	1	0.0065	10.27790	10.60026
8	10.50249	0.001	1	0.0075	10.31569	10.63001
9	10.52970	0.001	1	0.0085	10.34394	10.64659
10	10.54967	0.001	1	0.0095	10.37847	10.67958
11	10.58299	0.001	1	0.0105	10.43709	10.71669
12	10.61797	0.001	1	0.0115	10.44517	10.73739
13	10.63337	0.001	1	0.0125	10.47364	10.76505
14	10.64592	0.001	1	0.0135	10.50945	10.78972
15	10.67045	0.001	1	0.0145	10.53112	10.82021

The result array represents a sorted table of all of the results, and is used to produce the charts and statistics for the distribution. The table shows the value, the weight, the number of occurrences of the value, the cumulative probability of the value, and the 5% and 95% confidence bounds for the value at that cumulative probability. (Note, however, that confidence bounds cannot be computed if importance sampling has been used in the model.)

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively. If the column is too narrow to display the contents of a cell, holding the cursor over the cell displays a tool-tip with the contents.

Several additional points should be noted regarding Distribution Result Arrays:

- You can copy the contents of the table to the clipboard bypressing the cell in the upper left-hadn corner (#) to select the entire table, and then

pressing **Ctrl+C** to copy the contents to the clipboard**.** You can subsequently paste the table into another application (such as a spreadsheet).

- Values in the table, by definition, are sorted in ascending order. To sort in descending order, double-click on a column header twice. Double-click on a header again to return to ascending order.
- You can control the number of significant figures displayed in tables from the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).

  Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 524).

The manner in which the results array is created and used is discussed in detail in Appendix B.

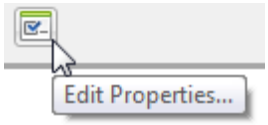
Viewing Distributions of Multiple Outputs

It is often useful to view the distributions of multiple outputs on a single chart or table. Displaying multiple distributions in this way allows you to more directly compare and contrast the results.

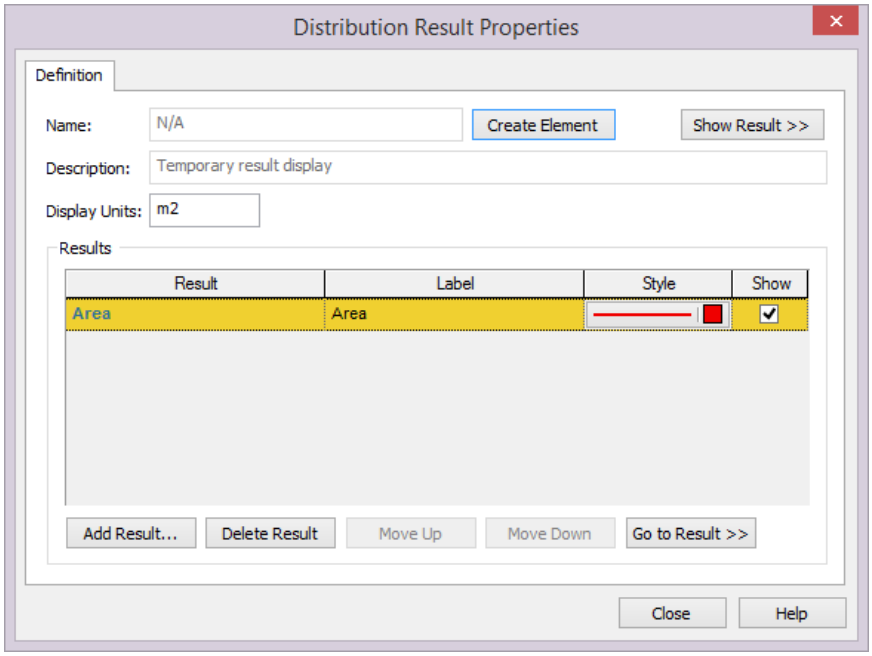
To display multiple results in a Distribution Result, you must first open the Result Properties dialog for the distribution.

Read more: Viewing the Properties of a Distribution Result (page 597).

You can do this by pressing the **Edit Properties…** button in any of the Distribution Result display windows:

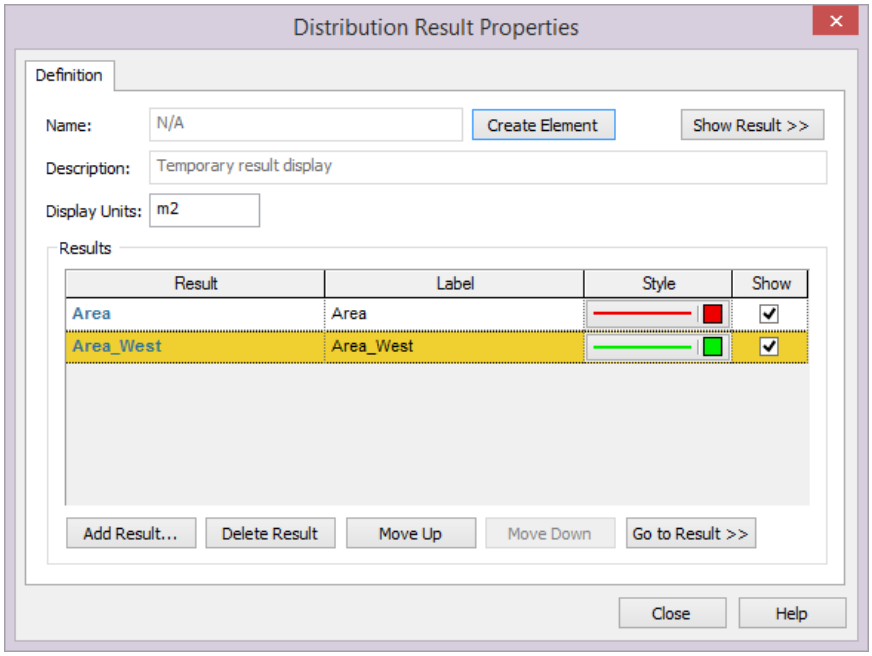


The following dialog will be displayed:



This is the dialog for an interative result. A Result element will have a Name defined. Note that the Label is user-editable and is used in legends and column headers.

To add addition outputs to the result, press the **Add Result…** button. A dialog for selecting an output will be displayed. After you select the output you wish to add to the result display, and press **OK**, the selected outputs are then shown in the Result Properties dialog:



You can show the full path for each result (showing the containment hierarchy) by placing your cursor over the Result item to display a tool-tip.

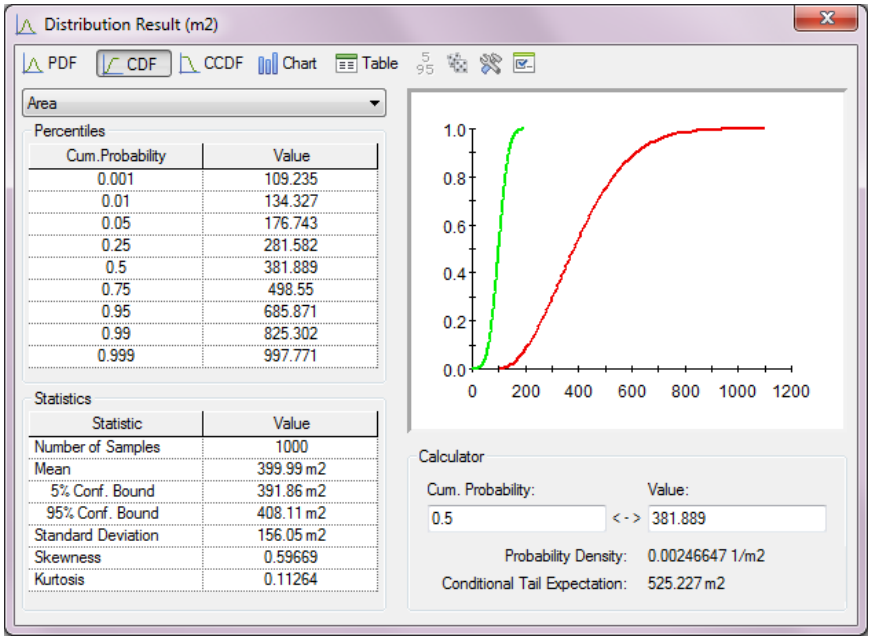
The following points should be noted regarding the Distribution Result Properties dialog when displaying multiple results:

- The units of the results default to the Display Units of the first result added to the list. However, you can change the **Display Units** that are displayed from within the Result Properties dialog.
- You can add as many results as desired to list. Note, however, that all results must have the same dimensions. GoldSim will not allow you to add an output which does not match the dimensions of the existing result(s).
- Only those results in which the **Show** box is checked will be included in displays. Hence, after adding a result to the list, you can temporarily hide it from displays by clearing this box.
- The **Label** is user-editable, and is used in legends and column headers.
- The Style of each line displayed in a Distribution Chart can be edited by clicking on the field (in the **Style** column) corresponding to each result.

Read more: Controlling the Chart Style in Distribution Results (page 626).

Pressing the **Show Result** button displays the multiple output display.

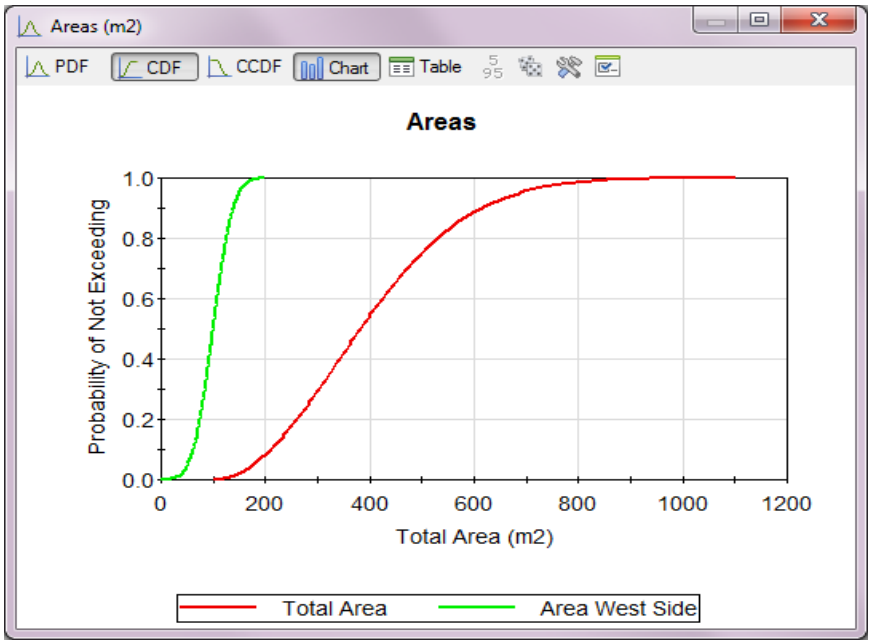
The Distribution Summary looks like this:



Note that although the Preview Pane will show all results, the Distribution Summary can only show Percentiles and Statistics for one result at a time. You can choose which of the results to display from the drop list at the upper left-hand corner of the window.

Read more: Viewing a Distribution Summary (page 600).

A Distribution Chart looks like this:



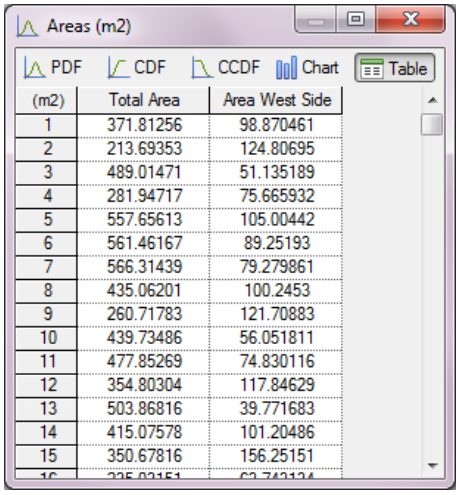
Note that the legend displays the (user-editable) Labels defined for each result.

Read more: Viewing a Distribution Chart (page 604).

Note that a legend is available for Distribution Charts (you may have to turn it on by right-clicking in the chart and selecting to show the legend from the

context menu). The Labels specified in the Result Properties dialog are used in the legend to label the different results.

For Distribution Tables, the results are listed in separate columns:



(m2)	Total Area	Area West Side
1	371.81256	98.870461
2	213.69353	124.80695
3	489.01471	51.135189
4	281.94717	75.665932
5	557.65613	105.00442
6	561.46167	89.25193
7	566.31439	79.279861
8	435.06201	100.2453
9	260.71783	121.70883
10	439.73486	56.051811
11	477.85269	74.830116
12	354.80304	117.84629
13	503.86816	39.771683
14	415.07578	101.20486
15	350.67816	156.25151

Note that the column headers are the (user-editable) Labels defined for each result.

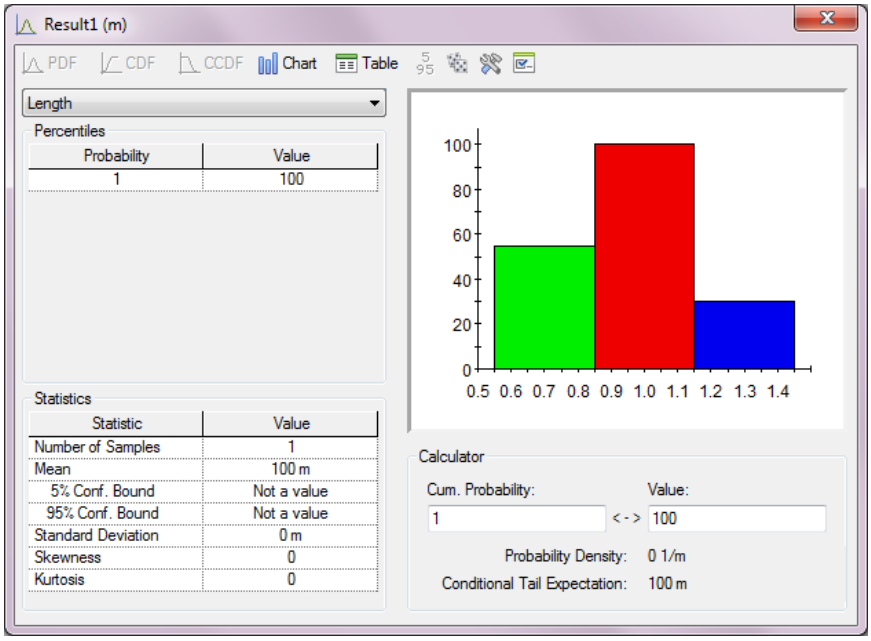
Read more: Viewing a Distribution Table (page 608).

Note: You cannot show Confidence Bounds in the Distribution Chart, Table or the Summary when displaying multiple outputs (the Confidence Bounds button is grayed out).

Viewing Distribution Results for Single Realization Runs

A Distribution Result can also be used ot view a single realization simulation. This is particularly useful if you want to compare different results (or scenarios) in a single display.

When displaying Distribution Results after running only a single realization, the Distribution Summary looks like this:



In this example, three results have been added to the Result via the Properties dialog.

Note that the Preview Pane will show all results as a bar chart. The Distribution Summary can only show Percentiles and Statistics for one result at a time. You can choose which of the results to display from the drop list at the upper left-hand corner of the window. In any case, however, for a single result, there are no Percentiles or Statistics to display (other than the single value).

Read more: Viewing a Distribution Summary (page 600).

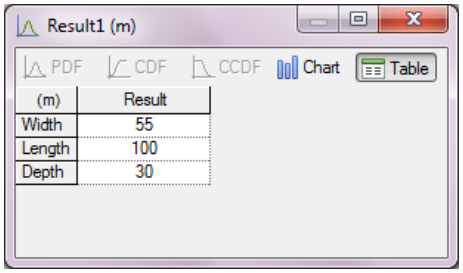
A Distribution Chart looks like this:

Note that the legend displays the (user-editable) Labels defined for each result.

Read more: Viewing a Distribution Chart (page 604).

A legend is available for Distribution Charts (you may have to turn it on by right-clicking in the chart and selecting to show the legend from the context menu). The Labels specified in the Result Properties dialog are used in the legend to label the different results.

For Distribution Tables, the results are listed in separate columns:



Note that the row headers are the (user-editable) Labels defined for each result.

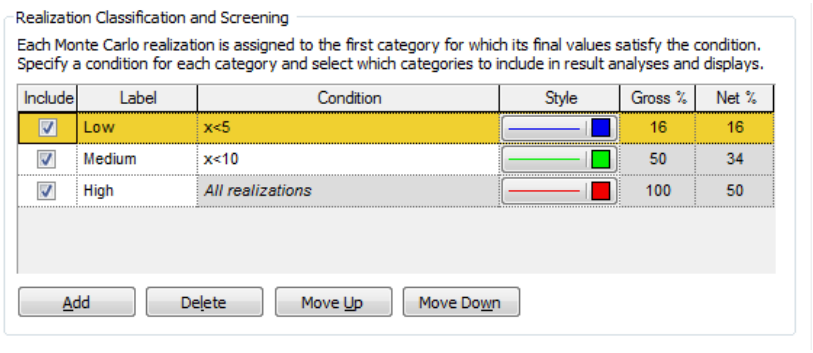
Read more: Viewing a Distribution Table (page 608).

Using Result Classification and Screening in Distribution Results

When carrying out probabilistic simulations, you may often run hundreds or thousands of realizations. In order to analyze the results, it is often quite useful to classify the realizations into *categories*. A category is simply defined by a condition relating one or more outputs in the model (e.g., those realizations in which the discount rate was above 3.5%; those realizations in which the profit exceeded $1,000,000; those realizations in which the peak concentration was between 1 mg/l and 10 mg/l).

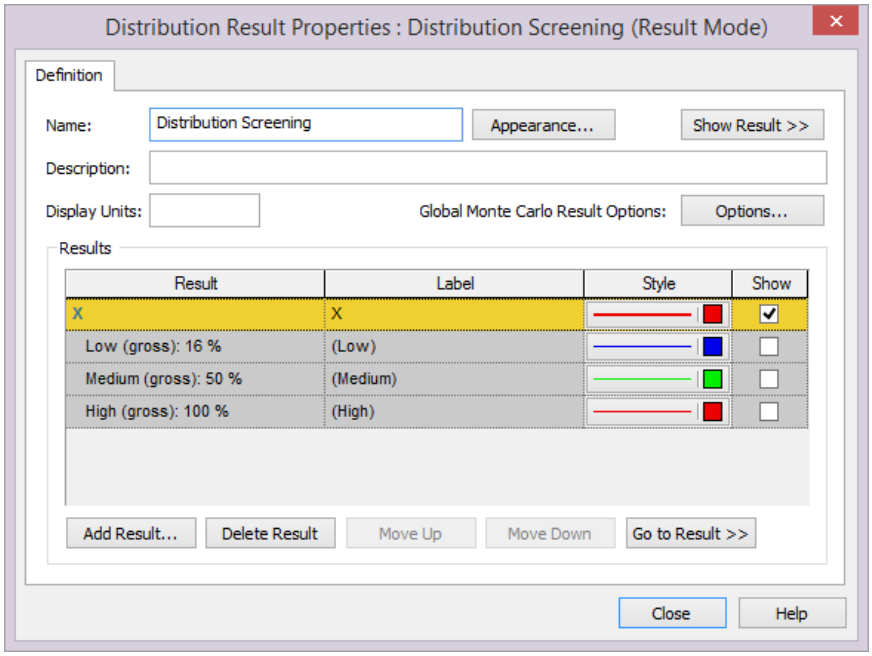
Read more: Classifying and Screening Realizations (page 533).

Classification categories are defined at the bottom of the Monte Carlo Result Display Properties dialog:



This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the **Result Options…** button), and is also accessible by pressing the **Options…** button available in the Result Properties dialog of a Distribution Result element.

Once you have defined categories, Distribution results can be displayed by category. The Properties dialog for a Distribution Result with categories defined looks like this:



In addition to showing the result (X in this case), the three categories that have been defined are also shown.

Note: Categories are only shown if the Distribution Result contains single result. If you add multiple results, the categories will not be shown.

By default, the categories are not shown in result displays (the **Show** box is cleared). If you want to display the categories, you must check the box.

In this particular example, the net number of realizations falling into each category is as follows: 16% of the realizations fall into the Low category, 34% fall into the Medium category, and 50% fall into the High category. This information is displayed in the the Monte Carlo Result Display Properties dialog after you run the simulation.

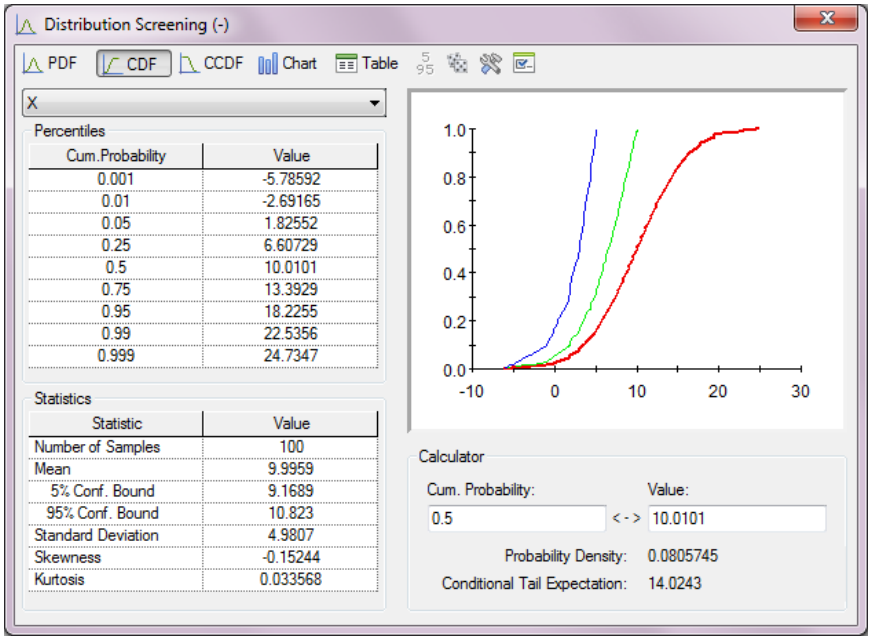
When you display a Distribution Result with categories, however, the results are displayed in terms of the gross number (as indicated in the Result Properties dialog). The gross number represents the percentage of realizations falling into the selected category and all the categories above it. Hence, in this example, 16% of the realizations fall into the Low category, 50% fall into the Low or Medium category, and 100% fall into the Low, Medium or High category.

If you check the **Show** boxes for all categories, and press the **Show Result** button, the categories will be included in the displays.

Note that for each category, you can edit the **Style** via the Monte Carlo Result Display Properties dialog. (The Style selections for each category are displayed in the Distribution Result Properties dialog, but cannot be edited there.)

Read more: Controlling the Chart Style in Distribution Results (page 626).

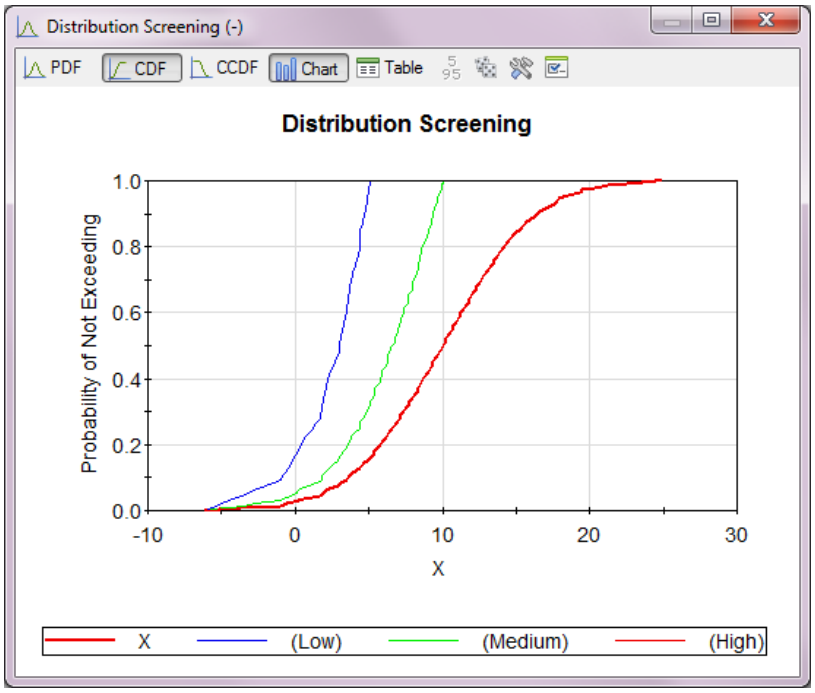
The Distribution Summary with categories looks like this:



Note that although the Preview Pane will show all categories, the Distribution Summary can only show Percentiles and Statistics for one category at a time. You can choose which of the categories to display from the drop list at the upper left-hand corner of the window.

Read more: Viewing a Distribution Summary (page 600).

A Distribution Chart with categories looks like this:



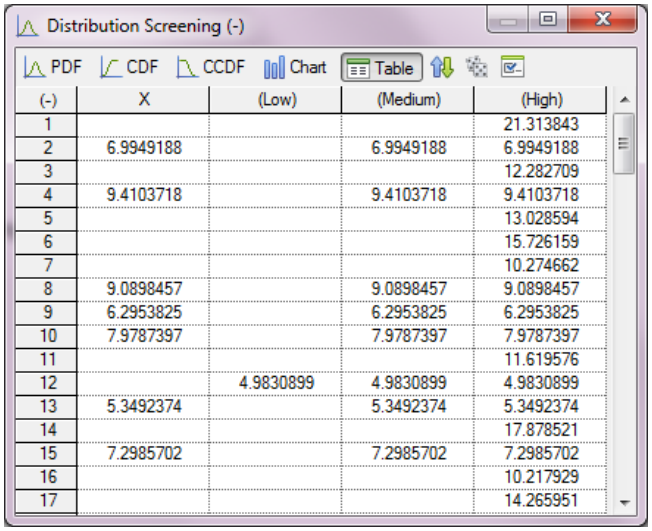
The Labels specified in the Monte Carlo Result Display Properties dialog are used in the legend to label the category results. The label for the result itself (in this case X) is specified in the Result element dialog.

Note that a legend is available for Distribution Charts (you may have to turn it on by right-clicking in the chart and selecting to show the legend from the context menu).

Note that because a Distribution Result displays categories in terms of gross numbers, the final category (in this case High) actually includes all realizations, and hence is identical the the display of the result itself.

Read more: Viewing a Distribution Chart (page 604).

For Distribution Tables with categories, the results are listed in separate columns:



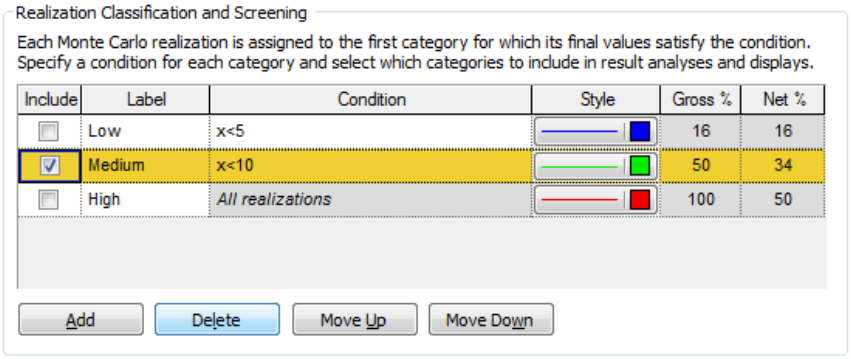
(-)	X	(Low)	(Medium)	(High)
1				21.313843
2	6.9949188		6.9949188	6.9949188
3				12.282709
4	9.4103718		9.4103718	9.4103718
5				13.028594
6				15.726159
7				10.274662
8	9.0898457		9.0898457	9.0898457
9	6.2953825		6.2953825	6.2953825
10	7.9787397		7.9787397	7.9787397
11				11.619576
12		4.9830899	4.9830899	4.9830899
13	5.3492374		5.3492374	5.3492374
14				17.878521
15	7.2985702		7.2985702	7.2985702
16				10.217929
17				14.265951

Note that only realizations that fall into each category are displayed. If a realization is not in a category, it is blank. Because a Distribution Result displays categories in terms of gross numbers (and categories are displayed in the table from left to right), if a realization falls into a category, it also falls into all category columns to the right.

Read more: Viewing a Distribution Table (page 608).

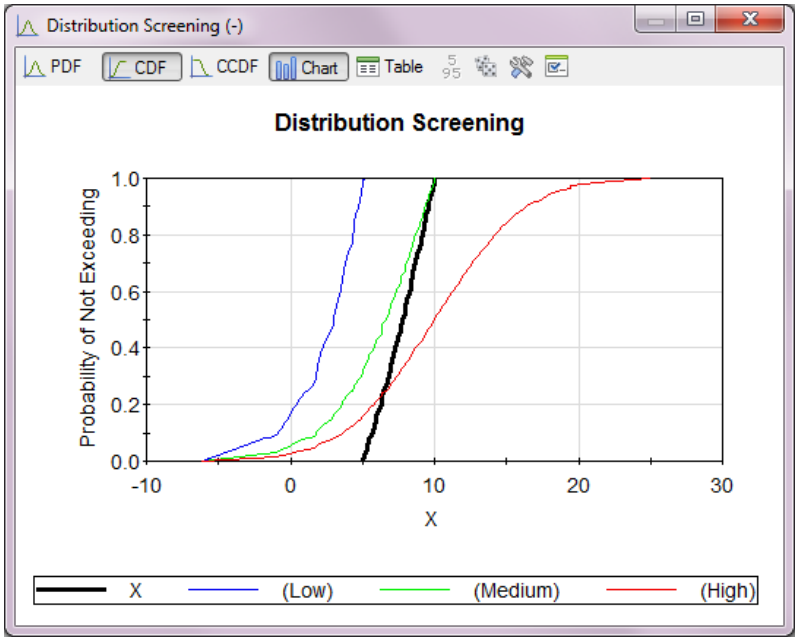
Note: You cannot show Confidence Bounds in the Distribution Chart, Table or the Summary when displaying categories (the Confidence Bounds button is grayed out).

Within all result displays, you can also choose to ***screen*** out one or more categories, so that the results that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include. You can screen a particular category from the results by clearing the **Include** box at the bottom of the Monte Carlo Result Display Properties dialog:



Note: You can edit the **Include** box in Edit Mode, Result Mode Scenario Mode or Run Mode.

Result screening only affects the result itself (if you choose to show categories, the screening does not apply to the category displays). Hence, in the example below, we are screening Low and High categories from the result (labeled X), but simultaneously displaying the (gross) category results:



Screened results shown with category results. In most cases, you would likely not show category results when screening.

Note that when screening results, it is the net number of realizations that are screened. Hence, in the example above (in which the Low and High categories are screened), the screened result is based on only those results in the Medium category (34% of the realizations).

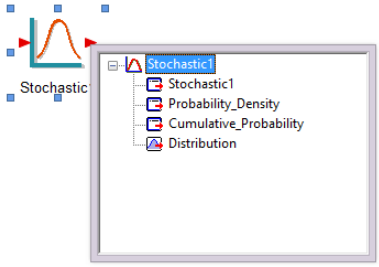
Adding a Distribution Output to a Distribution Result

In most cases, when you add an output to a Distribution Result (via the **Add Result…** button in the Result Properties dialog), you will simply be adding a standard output (such as the output of an Expression element or the the primary output of a Reservoir element). However, several types of elements in GoldSim produce a specialized output referred to as a Distribution output, and this output type can also be added (and viewed) in a Distribution Result.

Distribution outputs are complex outputs that represent all the statistical information necessary to define a probability distribution. They can ony be produced by Stochastic elements, SubModels and Spreadsheet elements.

Read more: Stochastic Elements (page 156); Creating the Output Interface to a SubModel (page 925); Importing Stochastic Element Definitions from a Spreadsheet (page 860).

These outputs can be clearly identified when viewing the output port of an element:



Note the the Distribution output has a different icon than the other three standard outputs.

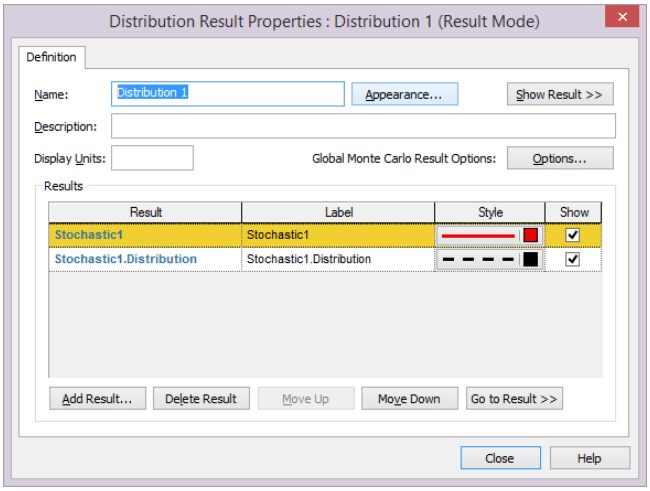
In some situations, you may want to add such an output to a Distribution Result. There are two primary cases where this can be of value:

1. You wish to carry out a nested Monte Carlo simulation (using a SubModel). In order to view the results, you would add a Distribution output from a Monte Carlo SubModel to a Distribution Result element in the parent model.

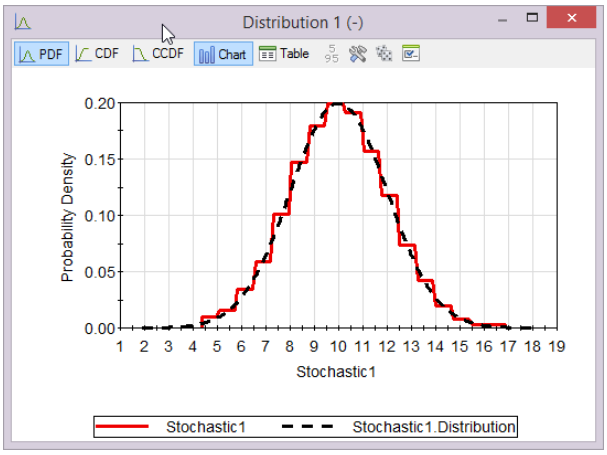
   Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933).

2. You may want to compare a sampled distribution to a specified analytical distribution (e.g., to see how closely the sampled distribution matches a particular distribution such as a Normal or Weibull).

   This is possible because Stochastic elements always have a Distribution output and if you view the Distribution output for a Stochastic in a Distribution Result it will display the analytical distribution (as opposed to the sampled distribution for that Stochastic). In the example below, a Stochastic element is defined (as a Normal) and the model is run for 300 realizations. Two of its outputs are added to a Distribution Result: the primary output and the Distribution output:



The Result display would look like this:



The primary output (Stochastic1) shows the sampled distribution. The Distribution output (Stochastic1.Distribution) shows the analytical distribution shape.

Viewing Scenario Results in Distribution Result Elements

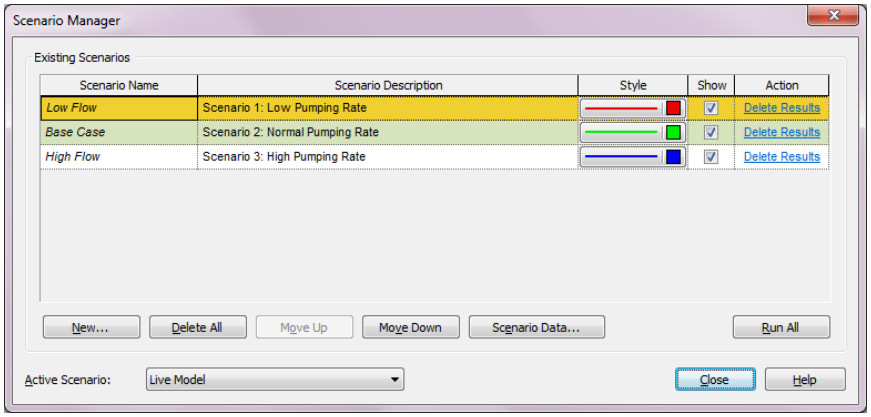
GoldSim's scenario modeling capability allows you to directly compare results generated by different sets of input parameters. In effect, when you use this capability, your model can store (and subsequently compare) multiple sets of results (and inputs).

Read more: Creating, Running and Comparing Scenarios (page 463).

When a model is in Scenario Mode, Distribution Result elements can be used to view scenario results.

Note: Scenario results are only shown for the *first result* listed in the Distribution Result element. If the Distribution Result contains multiple results, in Scenario Mode only the first result will be displayed.

In order for scenario results to be displayed in a Distribution Result element, the **Show** button must be checked for each scenario to be displayed in the Scenario Manager (accessed from the main menu (**Run | Scenario Manager…**) or by pressing **F7**):



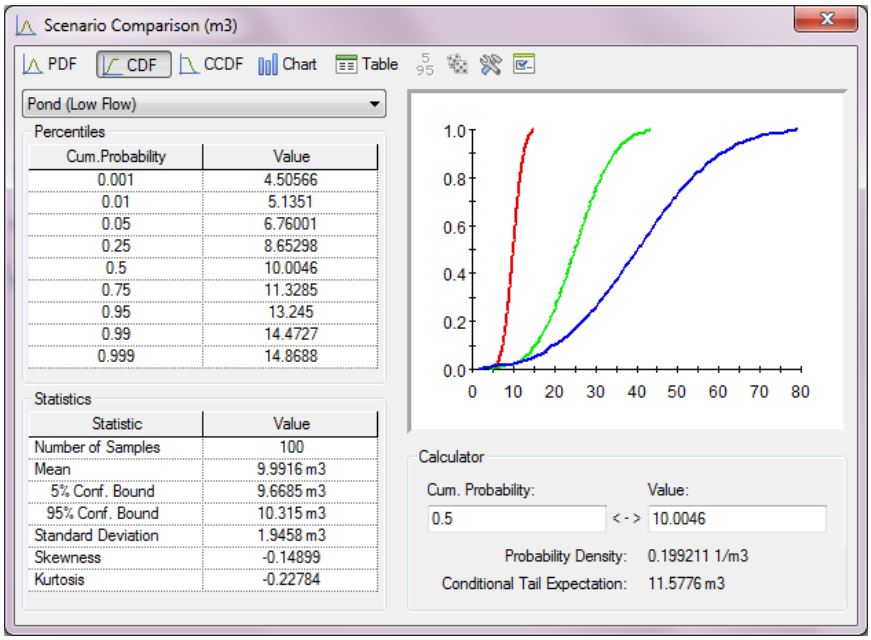
In this example, the Show box is checked for all three scenarios, so all three will be displayed in results.

If you double-click on a Distribution Result element in Scenario Mode, GoldSim displays results for all scenarios for which scenario results have been generated (and for which the **Show** button has been checked from within the Scenario Manager).

Note that for each scenario, within the Scenario Manager you can edit the **Style**, as well as the **Scenario Name**. These affect how the results are labled in chart and table displays.

Read more: Controlling the Chart Style in Distribution Results (page 626).

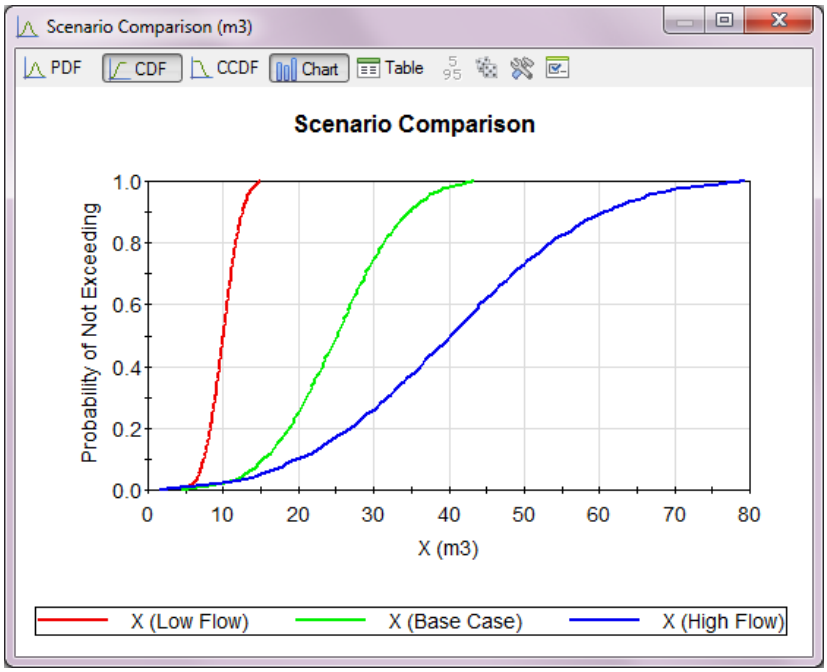
If you have run multiple realizations for the various scenarios (and hence are in Scenario Mode), a Distribution Summary in Scenario Mode looks like this:



Note that although the Preview Pane will show all scenarios, the Distribution Summary can only show Percentiles and Statistics for one scenario at a time. You can choose which of the scenarios to display from the drop list at the upper left-hand corner of the window.

Read more: Viewing a Distribution Summary (page 600).

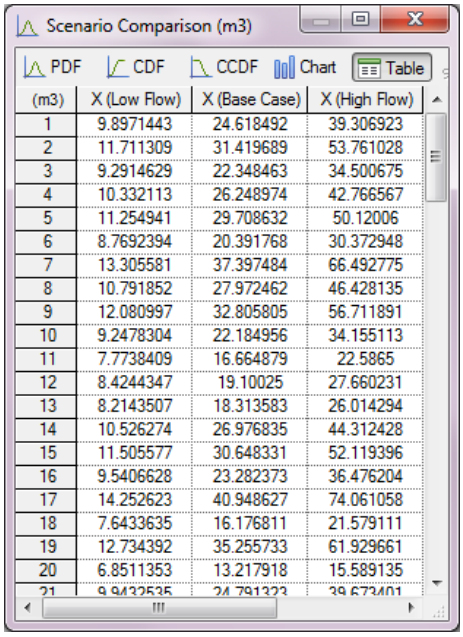
A Distribution Chart in Scenario Mode looks like this:



The Scenario Names specified in the Scenario Manager dialog are used in the legend (in parentheses) after the Result Label (specified in the Result element dialog).

Read more: Viewing a Distribution Chart (page 604).

For Distribution Tables in Scenario Mode, the scenarios are listed in separate columns:



(m3)	X (Low Flow)	X (Base Case)	X (High Flow)
1	9.8971443	24.618492	39.306923
2	11.711309	31.419689	53.761028
3	9.2914629	22.348463	34.500675
4	10.332113	26.248974	42.766567
5	11.254941	29.708632	50.12006
6	8.7692394	20.391768	30.372948
7	13.305581	37.397484	66.492775
8	10.791852	27.972462	46.428135
9	12.080997	32.805805	56.711891
10	9.2478304	22.184956	34.155113
11	7.7738409	16.664879	22.5865
12	8.4244347	19.10025	27.660231
13	8.2143507	18.313583	26.014294
14	10.526274	26.976835	44.312428
15	11.505577	30.648331	52.119396
16	9.5406628	23.282373	36.476204
17	14.252623	40.948627	74.061058
18	7.6433635	16.176811	21.579111
19	12.734392	35.255733	61.929661
20	6.8511353	13.217918	15.589135
21	9.9432535	24.791323	39.673401

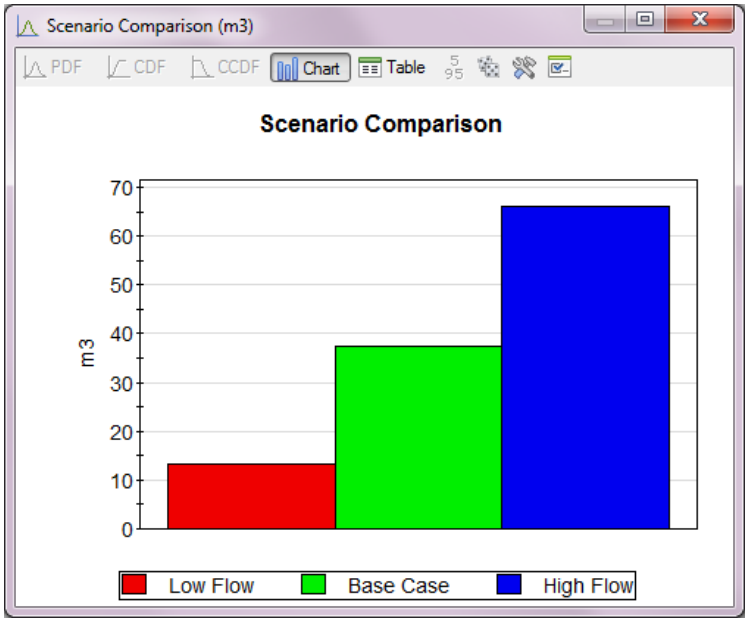
Read more: Viewing a Distribution Table (page 608).

Note: You cannot show Confidence Bounds in the Distribution Chart, Table or the Summary in Scenario Mode (the Confidence Bounds button is grayed out).

When running and comparing scenarios, it is often useful to view Distribution Results even when running only a single realization. This is because when displaying Distribution Results after running only a single realization, GoldSim will display the single results for each scenario side-by-side in the form or a bar chart.

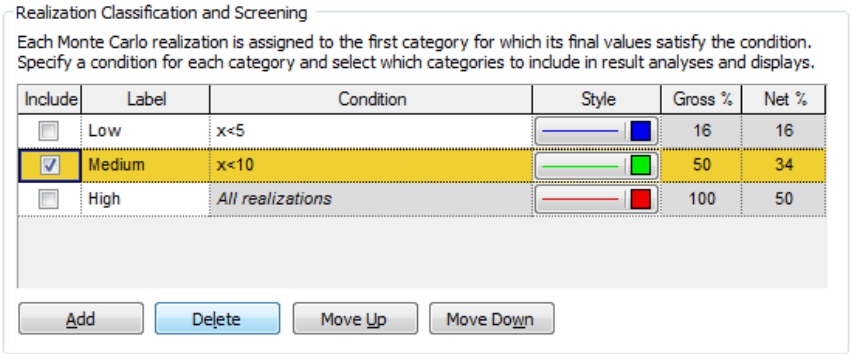
Read more: Viewing Distribution Results for Single Realization Runs (page 614).

For example, a Distribution Chart in Scenario Mode looks like this:



This particular chart shows three scenarios.

When viewing a Distribution Result in Scenario Mode, you cannot view results by category (i.e., based on the categories you may have defined at the bottom of the Monte Carlo Result Display Properties dialog):



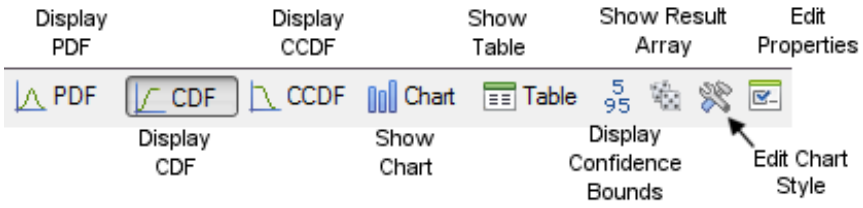
That is, in Scenario Mode, results are never displayed by category. However, screening by category is active in Scenario Model. That is, if you have chosen to ***screen*** out one or more categories (by clearing the **Include** box in the dialog above), the scenario results that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include.

Note: Depending on how you have defined the categories, it is possible for a realization to fall into a different category in different scenarios. Hence, when screening results in Scenario Mode, the number of realizations within each scenario may differ.

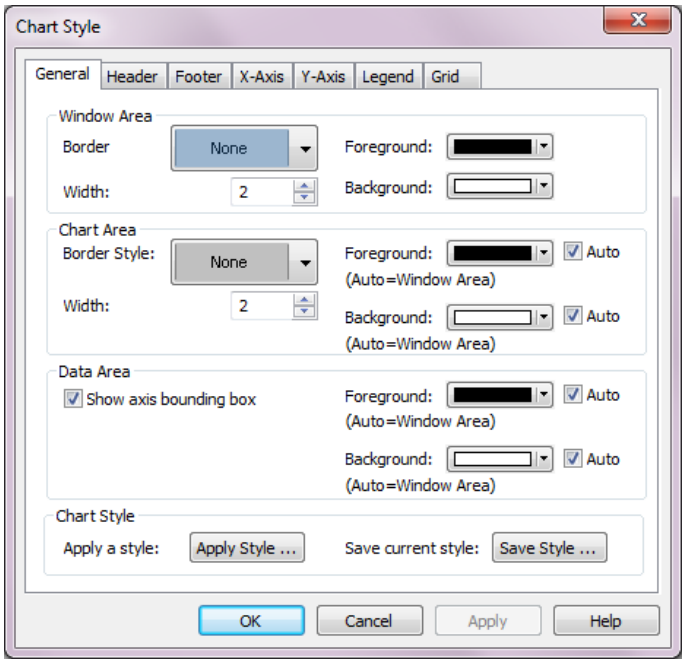
Read more: Using Result Classification and Screening in Distribution Results (page 616).

Controlling the Chart Style in Distribution Results

GoldSim has powerful charting capabilities that allow you to customize the appearance of each chart. This includes adding headers and footers, and changing axis scales and labels. Most of these attributes can be edited by pressing the **Edit Chart Style** button at the top of the Distribution Chart window:



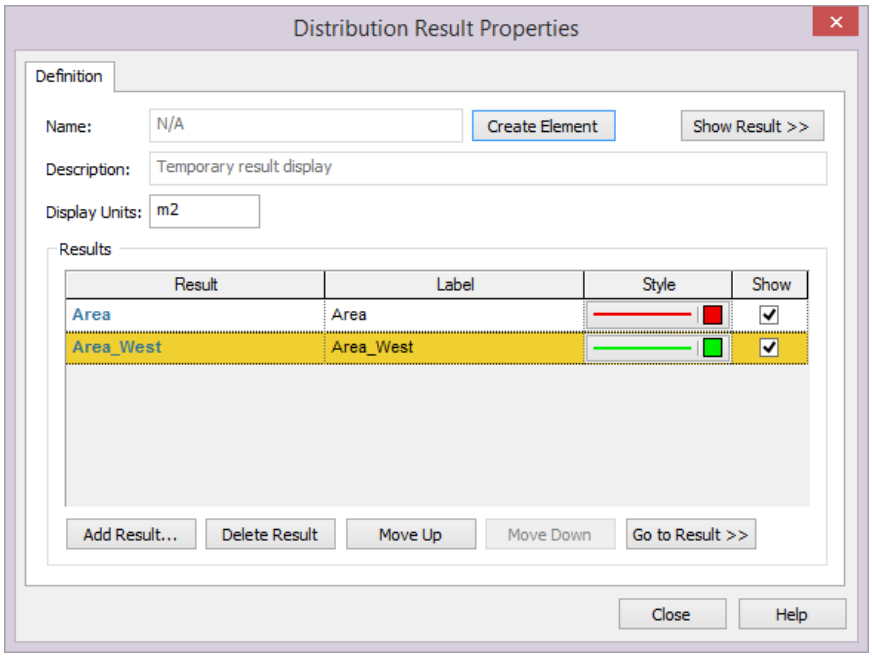
Pressing this button (or right-clicking in a chart and selecting **Edit Chart Style…**) provides access to the following dialog for editing the various chart properties:



This dialog is common to all types of charts, and is discussed elsewhere.

Read more: Editing the Appearance of a Chart (page 661).

In addition to the basic chart attributes controlled by the Chart Style dialog, the Distribution Result Properties dialog itself is used to control some of the attributes of a chart:

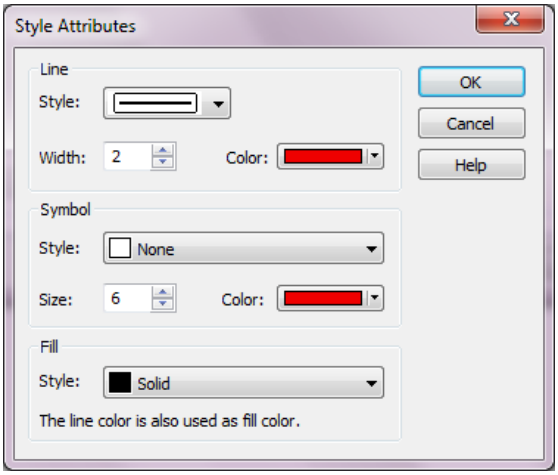


In particular,

- The units of the results default to the Display Units of the first result added to the list. However, you can change the **Display Units** that are displayed from within the Result Properties dialog.
- Only those results in which the **Show** box is checked will be included in displays. Hence, after adding a result to the list, you can temporarily hide it from displays by clearing this box.
- The **Label** is user-editable, and is used in legends (and column headers for tables).

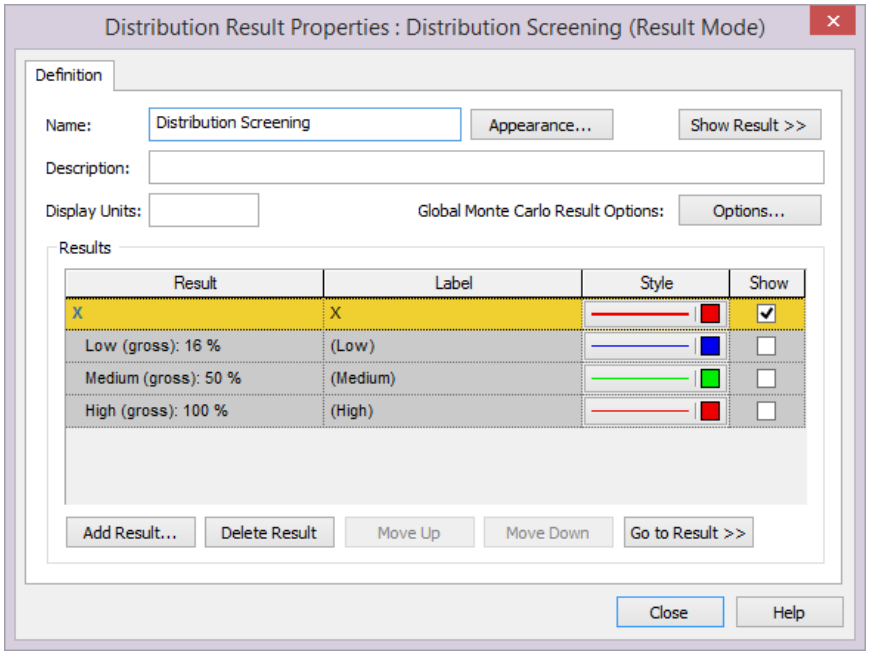
Note: You can hide or show the legend on a chart via the context menu (i.e., accessed by right-clicking in the chart).

- The Style of each line displayed in a Distribution Chart can be edited by clicking on the field (in the **Style** column) corresponding to each result. This provides access to the following dialog for editing the line style (for a distribution) or the fill style (for a single realization display):



Read more: Editing Data Styles (page 669).

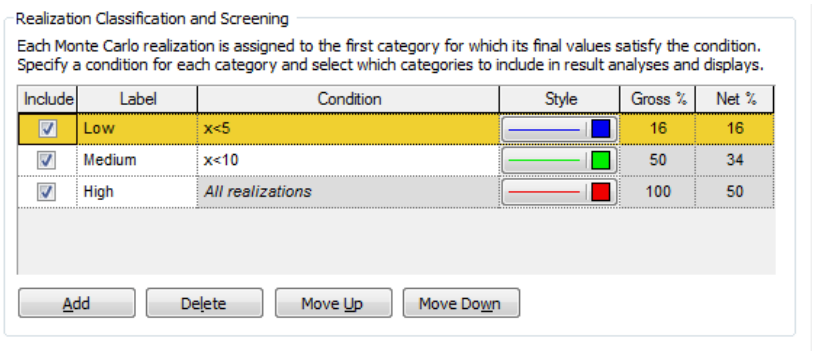
If you have defined realization categories, have run multiple realizations, and a Distribution Result element has only a single result defined, you will note that in addition to showing the result (X in the example below), any categories that have been defined are also shown:



Read more: Using Result Classification and Screening in Distribution Results (page 616).

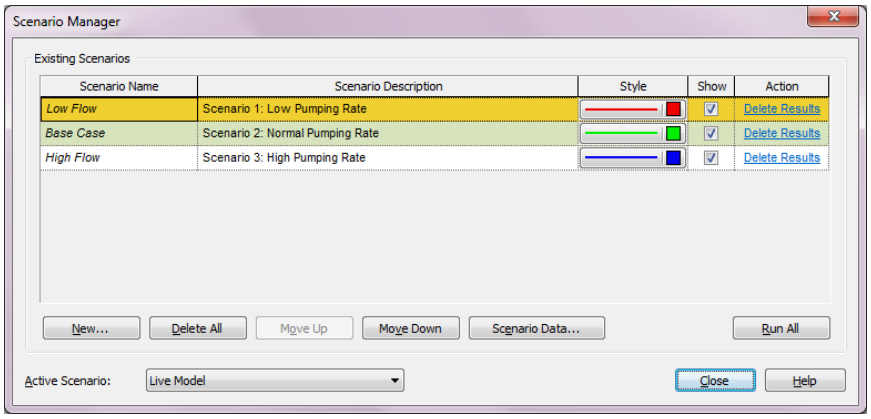
By default, the categories are not shown in result displays (the **Show** box is cleared). If you want to display the categories, you must check the box.

Note that although the Style selections for each category are displayed in the Distribution Result Properties dialog, they cannot be edited there (if you try to do so, you will note that they are grayed out). Instead, the **Style** for each category is controlled via the Monte Carlo Result Display Properties dialog (most easily accessed by pressing the **Options…** button in the Result Properties dialog of a Distribution Result element). The bottom of the Monte Carlo Result Display Properties dialog looks like this:



The Style used for each category displayed in a Distribution Chart can be edited by clicking on the field (in the **Style** column) corresponding to each category.

When a model is in Scenario Mode, Distribution Result elements can be used to view scenario results. In this case, the line style for distribution results for the different scenarios is controlled by the Scenario Manager dialog (accessed from the main menu (**Run | Scenario Manager…**) or by pressing **F7**):



Read more: Viewing Scenario Results in Distribution Result Elements (page 622).

In order for scenario results to be displayed in a Distribution Result element, the **Show** button must be checked for each scenario to be displayed in the Scenario Manager.

Note that for each scenario, you can edit the **Style**, as well as the **Scenario Name**, which is used used in legends (and column headers for tables).

Viewing Multi-Variate Results

Multi-Variate results allow you to analyze and compare multiple outputs in graphical or tabular form. That is, by definition, a Multi-Variate result display requires at least two separate output variables (i.e., results). A Multi-Variate result display provides a mechanism for creating scatter plots, as well as carrying out sensitivity and correlation analysis among the selected outputs.

To view Multi-Variate results, you must select a specific output that you are interested in, along with one or moe additional outputs that may have affected that result.

Five types of Multi-Variate result displays are available:

- 2D scatter plots;

- 3D scatter plots;
- Sensitivity analyses;
- Correlation analyses; and
- "Raw" Data display.

Note: Multi-Variate results are not available in Scenario Mode. They can only be viewed in Result Mode.

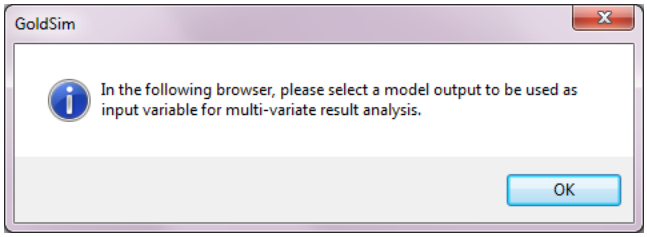
Read more: Understanding Simulation Modes (page 456).

Selecting Outputs for a Multi-Variate Result Display

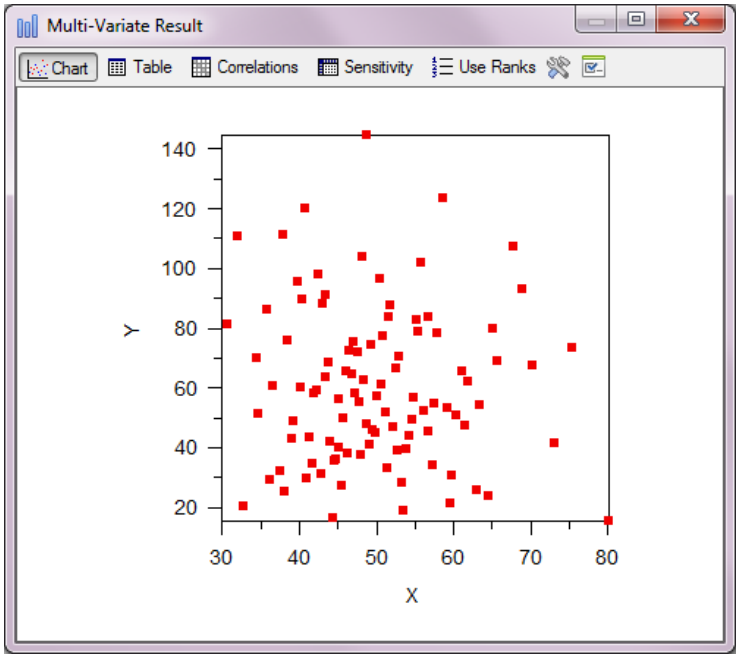
Unlike other result types, a Multi-Variate results requires specification of at least two outputs. If you have saved Final Values for an output, you can display a Multi-Variate result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Multi-Variate Result…** from the context menu.

Read more: Saving Outputs as Results (page 453).

The output that you selected will be defined as the result to evaluate. You will then be prompted to select an input variable (i.e., some other output in the model) upon which you wish to base your analysis (multi-variate result displays require at least one variable to be selected):



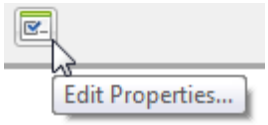
A browser will then appear allowing you to select a variable (i.e., another output in the model). After doing so, the default display for a Multi-Variate result (a 2D scatter plot) is displayed:



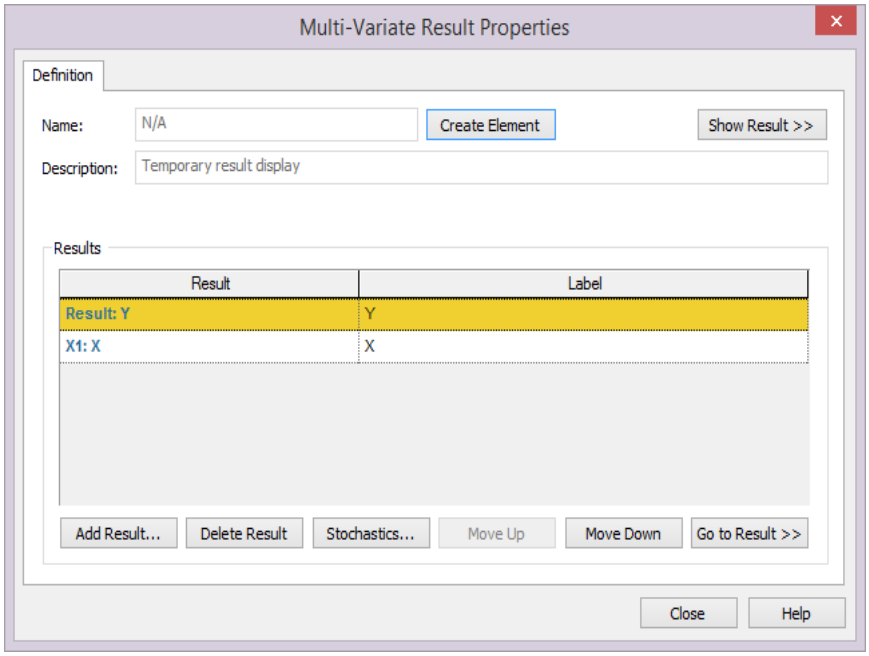
Note: By default, a legend will be turned on, labeling the different categories into which the realizations have been classified. If you have not defined categories, you will want to hide the legend (by right-clicking in the chart and and clearing **View | Show Legend**).

Read more: Using Result Classification and Screening in Multi-Variate Results (page 643).

To add additional outputs to the the Mulit-Variate result, you must then press the Result Properties button. The Result Properties button is the furthest button to the right at the top of the display window:



When you do so, the Result Properties dialog for the Mult-Variate result will be displayed:



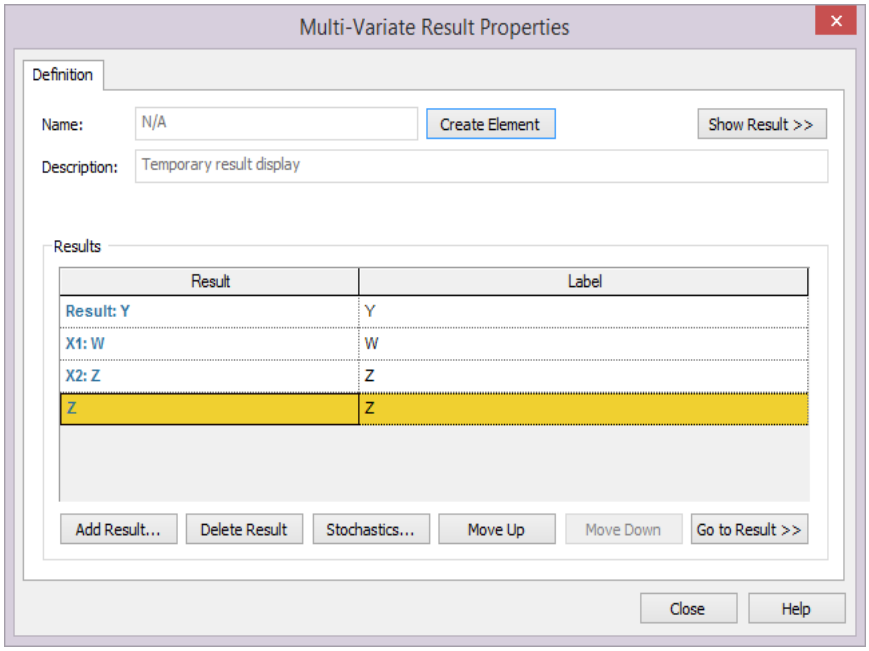
Read more: Viewing the Properties of a Multi-Variate Result (page 633).

From this dialog, you can add additional variables (or delete variables) using the **Add Result…** and **Delete Result** buttons.

Note: If you delete variables such that you have less than two, you will not be able to display any results at all (since a Multi-Variate result requires at least two variables).

The **Stochastics…** button adds all Stochastic elements in the model to the list of variables. This is often of value, since when carrying out sensitivyt or uncertaintly analyses, the variable of interest are typically the Stochastics in your model.

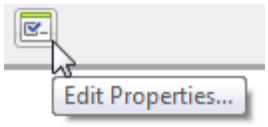
For the purpose of creating scatter plots, the variables must be assigned to specific axes. The result (the output from which you selected **Multi-Variate Result…**) is assigned to the Y-axis (i.e., the "Result" axis). The first variable that you select is assigned to the X1-axis, and the second variable you select is assigned to the X2-axis. Additional variables are not assigned to any axis. This is indicated in the Result column of the dialog:



You can subsequently reassign these axes by moving variables up and down in the list using the **Move Up** and **Move Down** buttons..

Viewing the Properties of a Multi-Variate Result

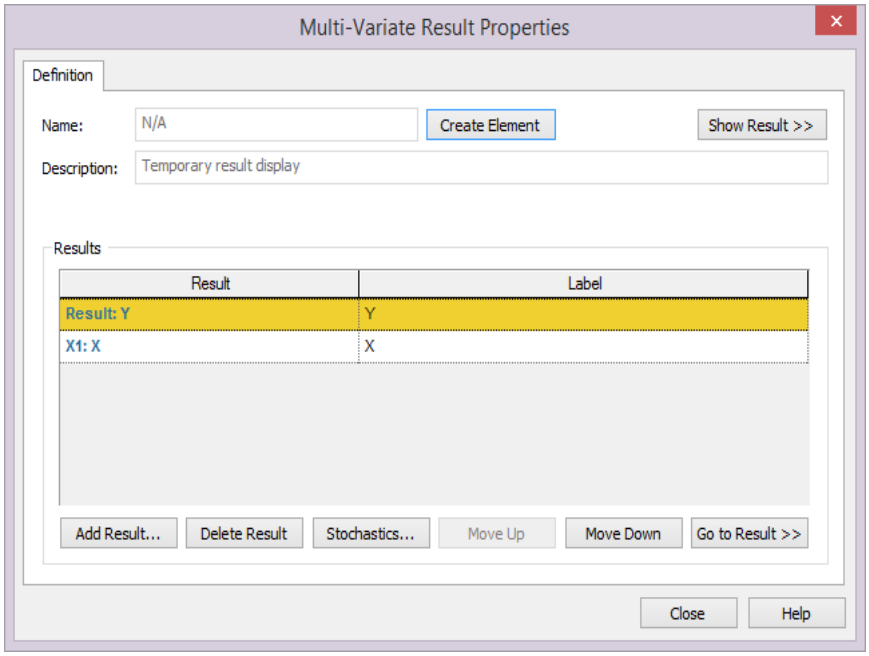
If you click on the Result Properties button when viewing a Multi-Variate Result chart or table, the Properties dialog for the result will be displayed. The Result Properties button is the furthest button to the right at the top of the display window:



Note that when you press this button, it does not close the result display window you were displaying. Both windows are displayed simultaneously (although you can subsequently close one or the other). When viewing results, any changes you make to the Result Properties are immediately represented in the result display.

The Result Properties dialog is always modal. That is, it cannot be minimized, and with the exception of viewing the result display (which can be displayed simultaneously and share the focus with the Properties dialog), it retains the focus while it is displayed. As a result, you must close the Result Properties dialog before you can edit any other part of your GoldSim model.

The Result Properties dialog for an interactive Multi-Variate result looks like this:



At the top of the page is a button to **Create Element**. By pressing this button, a Result element is created. When you do so, the key change that will be noticeable is that the **Name** will now be editable. The **Create Element** button becomes an **Appearance** button (which is used to modify the appearance of the element itself).

Read more: Creating and Using Result Elements (page 526).

The **Show Result >>** button opens a result display window (while keeping the Properties dialog simultaneously open.

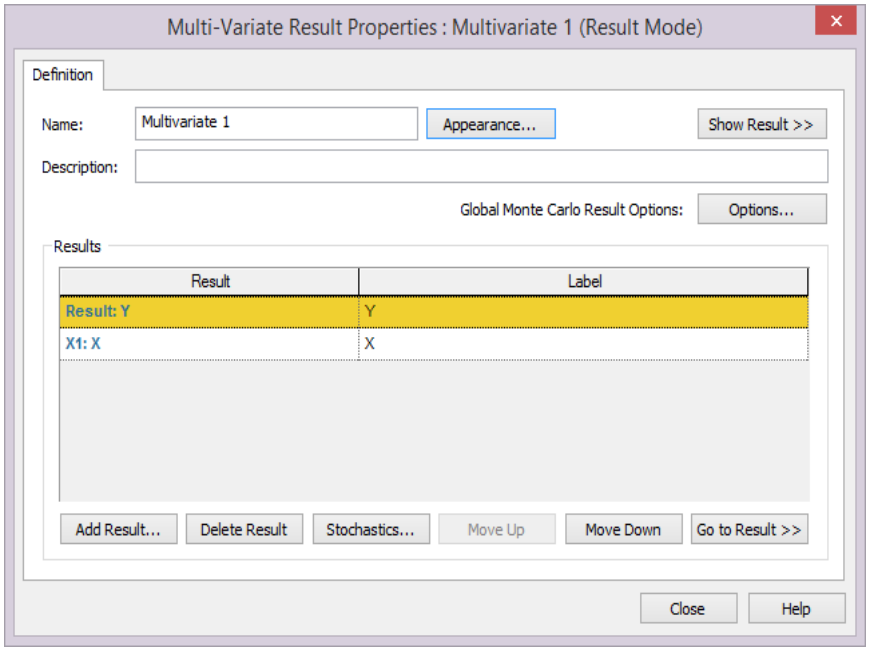
Read more: Viewing a 2D Scatter Plot (page 635); Viewing a 3D Scatter Plot (page 637); Viewing a Sensitivity Analysis Table (page 639); Viewing a Correlation Matrix Table (page 641); Viewing a Raw Multi-Variate Data Table (page 642).

The **Add Result…** button allows you to add other results to the chart. Results can be deleted with the **Delete Result** button.The order in which multiple results appear in the list can be changed using the **Move Up** and **Move Down** buttons. The **Stochastics…** button adds all Stochastic elements in the model to the list of variables. This is often of value, since when carrying out sensitivyt or uncertaintly analyses, the variable of interest are typically the Stochastics in your model.

For the purpose of creating scatter plots, the variables must be assigned to specific axes. The first result in the list is assigned to the Y-axis (i.e., the "Result" axis). The second result is assigned to the X1-axis, and the third result is assigned to the X2-axis. Additional variables are not assigned to any axis. This is indicated in the Result column of the dialog.

Read more: Selecting Outputs for a Multi-Variate Result Display (page 630).

The Properties dialog for a Multi-Variate Result element has several differences:



- The **Create Element** button is replaced by an **Appearance** button (which is used to modify the appearance of the element itself).
- The **Name** (and **Description**) are editable (since this is an element).
- An **Options…** button is available that provides access to the Monte Carlo Result Display Properties.

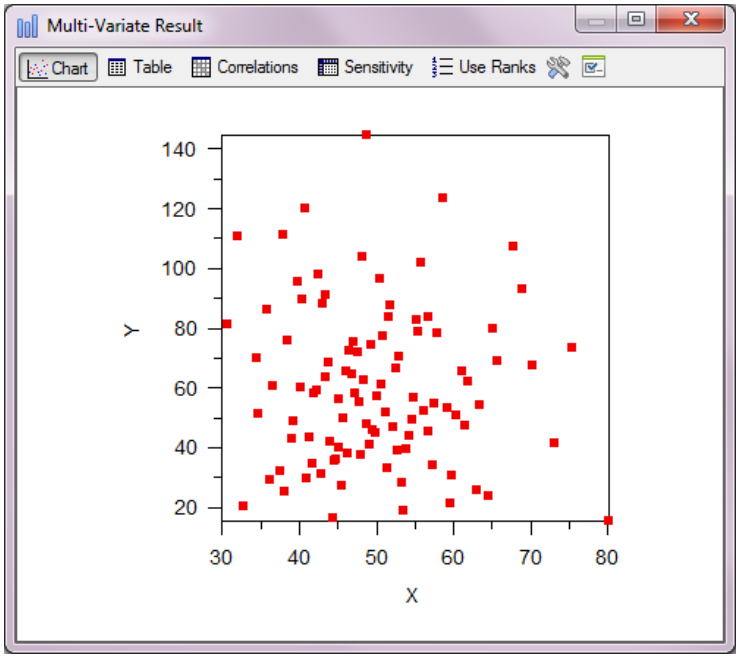
Read more: Controlling Monte Carlo Result Options (page 441).

Viewing a 2D Scatter Plot

A 2D Scatter Plot can provide a powerful visual image of the relationship between two variables.

If you have saved Final Values for an output, you can display a 2D Scatter Plot by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Multi-Variate Result…** from the context menu. The output that you selected will be defined as the result to evaluate. You will then be prompted to select an input variable (i.e., some other output in the model) upon which you wish to base your analysis (multi-variate result displays require at least one variable to be selected)

A browser will then appear allowing you to select a variable (i.e., another output in the model). After doing so, the default display for a Multi-Variate result (a 2D scatter plot) is displayed:



If you are viewing a different type of display (either a chart or table), you can view a 2D Scatter Plot by pressing the **Chart** or **2D Chart** button at the top of the display.

Note: If only two variables are listed in the Result Properties dialog, only a 2D Chart can be shwon, and hence only one button will be available (**Chart**). If three or more variables are present in the Result Properties dialog, buttons for both a **2D Chart** and a **3D Chart** become available.

Note: When viewing a Multi-Variate Result element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

For the purpose of creating scatter plots, the variables must be assigned to specific axes. The first result in the list of results in the Result Properties dialog is assigned to the Y-axis (i.e., the "Result" axis). The second result is assigned to the X1-axis, and the third result is assigned to the X2-axis. Additional variables are not assigned to any axis. This is indicated in the Result column of the Result Properties dialog. The order in which multiple results appear in the list can be changed using the **Move Up** and **Move Down** buttons in that dialog.

Read more: Selecting Outputs for a Multi-Variate Result Display (page 630).

The **Use Ranks** button determines whether the display uses ranks of the values or the actual values. By default, this button is not pressed, and the display uses actual values.

You can copy the chart to the clipboard by right-clicking in the chart, and selecting **Copy** from the context menu. Alternatively, you can press **Ctrl+C** while viewing the chart. To export a chart, right-click in the chart, and select **Export…** from the context menu.

Read more: Copying a Chart or Table (page 691); Exporting a Chart (page 691).

By default, Multi-Variate results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 525).

If you are using Result Classification, the plots points will be displayed in a different color for each category that has been defined.

Read more: Using Result Classification and Screening in Multi-Variate Results (page 643).

Viewing a 3D Scatter Plot

A 3D Scatter Plot (or a Cloud Plot) allows you to visualize the relationship between three variables.

The default view for a Multi-Variate result is a 2D Scatter Plot.

Read more: Viewing a 2D Scatter Plot (page 635).

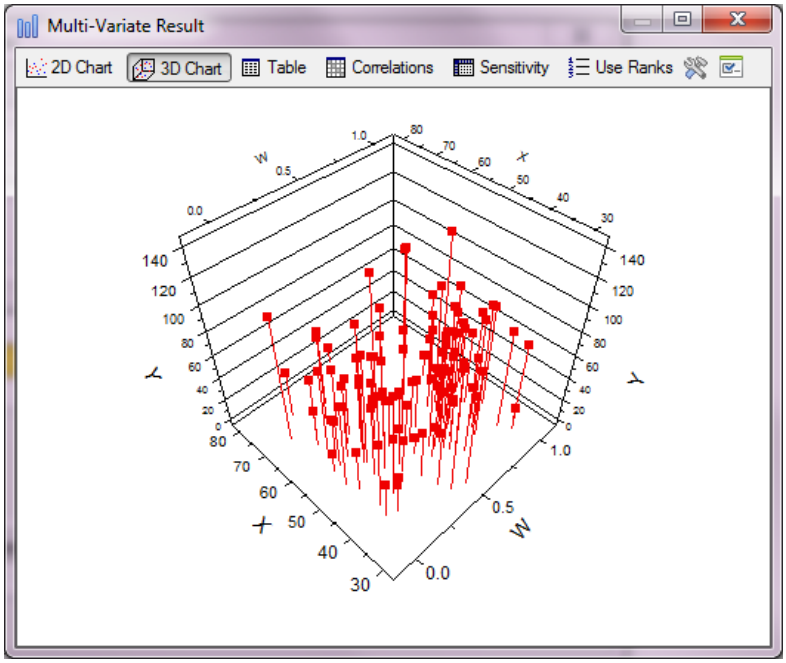
If you are viewing a 2D Scatter Plot (or a different type of Multi-variate display), you can view a 3D Scatter Plot by pressing the **3D Chart** button at the top of the display.

Note: Depending on how many variables you have defined in the Result Properties dialog, the 3D Chart button may not be available. In particular, in order to display a 3D Scatter Plot, you must have at least three results defined.

Read more: Viewing the Properties of a Multi-Variate Result (page 633).

Note: When viewing a Multi-Variate Result element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

A 3D Scatter Plot looks like this:



For the purpose of creating scatter plots, the variables must be assigned to specific axes. The first output in the list of results in the Result Properties dialog is assigned to the Y-axis (i.e., the "Result" axis). The second output is assigned to the X1-axis, and the third output is assigned to the X2-axis. Additional variables are not assigned to any axis. This is indicated in the Result column of the Result Properties dialog. The order in which multiple results appear in the list can be changed using the **Move Up** and **Move Down** buttons in that dialog.

Read more: Selecting Outputs for a Multi-Variate Result Display (page 630).

The **Use Ranks** button determines whether the display uses ranks of the values or the actual values. By default, this button is not pressed, and the display uses actual values.

You can rotate the three dimensional image by holding down the left mouse button and dragging within the chart. You can also use the Up, Down Left and Right cursor keys, or use the rotate options in the context menu for the chart.

In addition, 3D Scatter Plots have several hot-keys which can be used to modify the plot:

Keys	Action
Ctrl+D	Toggles the drop lines (the lines drawn from each point to the base of the axis) on and off.
Ctrl+S	Toggles between a 3D scatter plot and a 2D scatter plot.

These options are also available from the context menu for the chart.

You can copy the chart to the clipboard by right-clicking in the chart, and selecting **Copy** from the context menu. Alternatively, you can press **Ctrl+C** while viewing the chart. To export a chart, right-click in the chart, and select **Export…** from the context menu.

Read more: Copying a Chart or Table (page 691); Exporting a Chart (page 691).

By default, Multi-Variate results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 525).

If you are using Result Classification, the plots points will be displayed in a different color for each category that has been defined.

Read more: Using Result Classification and Screening in Multi-Variate Results (page 643).

Viewing a Sensitivity Analysis Table

One of the most powerful options for analyzing multi-variate results is to carry out a sensitivity analysis. GoldSim provides a number of statistical sensitivity analyses through the multi-variate result display option.

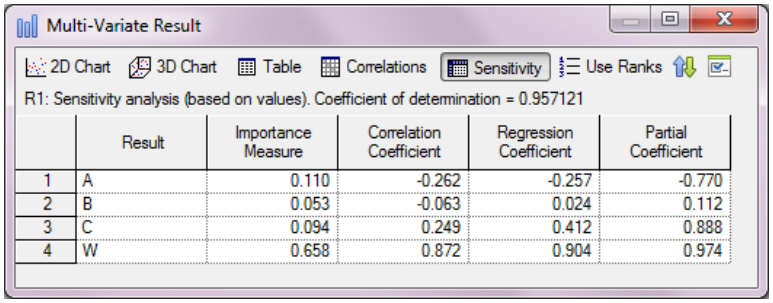
The default view for a Multi-Variate result is a 2D Scatter Plot.

Read more: Viewing a 2D Scatter Plot (page 635).

If you are viewing a 2D Scatter Plot (or a different type of Multi-variate display), you can view a Sensitivity Analysis Table by pressing the **Sensitivity** button at the top of the display.

Note: When viewing a Multi-Variate Result element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

A Sensitivity Analysis Table looks like this:



	Result	Importance Measure	Correlation Coefficient	Regression Coefficient	Partial Coefficient
1	A	0.110	-0.262	-0.257	-0.770
2	B	0.053	-0.063	0.024	0.112
3	C	0.094	0.249	0.412	0.888
4	W	0.658	0.872	0.904	0.974

This table displays measures of the sensitivity of the first output in the list of results in the Result Properties dialog (listed at the top of the dialog) to the selected input variables (all the other outputs in the list of results in the Result Properties dialog). In the example above, the analysis is being carried out on the result R1, and the other results whose impact on R1 is being measured are A, B, C and W.

Note: If some of your Stochastics are triggered repeatedly during a simulation, selecting them for sensitivity analysis is likely not to produce meaningful results.

The order in which results appear in the list in the Result Properties dialog can be changed using the **Move Up** and **Move Down** buttons in that dialog, such

that any variable in the list can be specied as the result for which the sensivitity analysis is being carried out.

Read more: Selecting Outputs for a Multi-Variate Result Display (page 630).

Each of the measures computed in the Sensitivity Analysis Table is described below:

Coefficient of determination: This coefficient varies between 0 and 1, and represents the fraction of the total variance in the result that can be explained based on a linear (regression) relationship to the input variables (i.e., Result = aX + bY + cZ + …). The closer this value is to 1, the better the relationship between the result and the variables can be explained with a linear model.

Importance Measure: This measure varies between 0 and 1, and represents the fraction of the result's variance that is explained by the variable. This measure is useful in identifying nonlinear, non-monotonic relationships between an input variable and the result (which conventional correlation coefficients may not reveal). The importance measure is a normalized version of a measure discussed in Saltelli and Tarantola (2002).

Correlation Coefficient: Rank (Spearman) or value (Pearson) correlation coefficients range between -1 and 1, and express the extent to which there is a linear relationship between the selected result and an input variable.

SRC (Standardized Regression Coefficient): Standardized regression coefficients range between -1 and 1 and provide a normalized measure of the linear relationship between variables and the result. They are the regression coefficients found when all of the variables (and the result) are transformed and expressed in terms of the number of standard deviations away from their mean. GoldSim's formulation is based on Iman et al (1985).

Partial Correlation Coefficient: Partial correlation coefficients vary between -1 and 1, and reflect the extent to which there is a linear relationship between the selected result and an input variable, after removing the effects of any linear relationships between the other input variables and both the result and the input variable in question. For systems where some of the input variables may be correlated, the partial correlation coefficients represent the "unique" contribution of each input to the result. GoldSim's formulation is based on Iman et al (1985).

A more detailed discussion of how these measures are computed (along with full references) is presented in Appendix B,.

You can copy the contents of a sensitivity analysis table to the clipboard. To do so, you must first select the entire table (by double-clicking on the empty cell in the upper left-hand corner of the table. After you do so, you can copy the table to the clipboard by **Ctrl+C**. You can subsequently paste the table into another application (such as a spreadsheet).

Like all result tables in GoldSim, this table can be sorted in ascending or descending order by selecting a column and pressing the **Sort** button.

Read more: Sorting Values in Result Tables (page 524).

The **Use Ranks** button determines whether the calculations use ranks of the values or the actual values. By default, this button is not pressed, and the calculations are based on actual values.

By default, Multi-Variate results operate on Final Values. That is, the analysis applies to the values at the end of each realization. However, by defining Capture Points, you can carry out the analysis at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to analyze.

Read more: Viewing Results at Capture Points (page 525).

Note: The sensitivity analyses presented here are statistical measures computed by analyzing multiple realizations of the model in which all of the Stochastic variables are simultaneously sampled each realization. GoldSim also provides a second type of sensitivity analysis in which you can vary one variable at a time, while holding all other variables constant.

Read more: Running Sensitivity Analyses (page 497).

Note: If you need to carry out more advanced sensitivity analyses, you can do so by exporting all results and using a third party analysis tool.

Read more: Exporting Results (page 678).

Viewing a Correlation Matrix Table

A Correlation Matrix tabulates the statistical correlation between two or more variables.

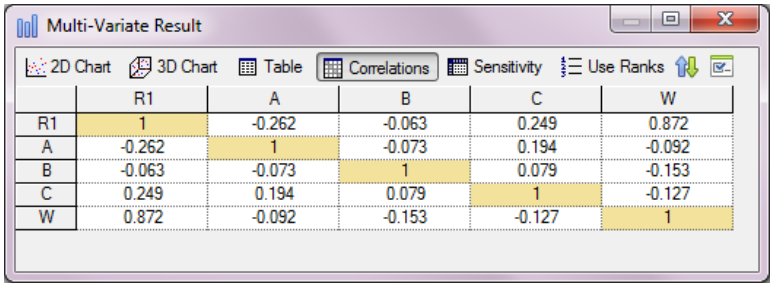
The default view for a Multi-Variate result is a 2D Scatter Plot.

Read more: Viewing a 2D Scatter Plot (page 635).

If you are viewing a 2D Scatter Plot (or a different type of Multi-variate display), you can view a Correlation Matrix Table by pressing the **Correlations** button at the top of the display.

Note: When viewing a Multi-Variate Result element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

A Correlation Matrix Table looks like this:



	R1	A	B	C	W
R1	1	-0.262	-0.063	0.249	0.872
A	-0.262	1	-0.073	0.194	-0.092
B	-0.063	-0.073	1	0.079	-0.153
C	0.249	0.194	0.079	1	-0.127
W	0.872	-0.092	-0.153	-0.127	1

The correlation matrix consists of a table whose column and row headers are identical, and consist of all of the selected variables. The table entry for a particular row/column pair is the correlation coefficient for that pair of variables. The correlation coefficient takes on a value between –1 and 1, 1 being a perfect positive correlation, -1 being a perfect negative correlation.

This is the same analysis presented in the Sensitivity Analysis Table (in the Correlation Coefficient column), but is presented in terms of a matrix here, showing how all variables are correlated to each of the others. The Sensitivity Analysis table simply shows how a single result variable is correlated to the others (i.e., it shows one column of the matrix).

Read more: Viewing a Sensitivity Analysis Table (page 639).

A detailed discussion of how correlation coefficients are computed is presented in Appendix B .

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively. If the column is too narrow to display the contents of a cell, holding the cursor over the cell displays a tool-tip with the contents.

You can copy the contents of a correlation matrix table to the clipboard. To do so, you must first select the entire table (by double-clicking on the empty cell in the upper left-hand corner of the table. After you do so, you can copy the table to the clipboard by **Ctrl+C**. You can subsequently paste the table into another application (such as a spreadsheet).

Like all result tables in GoldSim, this table can be sorted in ascending or descending order by selecting a column and pressing the **Sort** button.

Read more: Sorting Values in Result Tables (page 524).

The **Use Ranks** button determines whether the calculations use ranks of the values or the actual values. By default, this button is not pressed, and the calculations are based on actual values.

By default, Multi-Variate results operate on Final Values. That is, the analysis applies to the values at the end of each realization. However, by defining Capture Points, you can carry out the analysis at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to analyze.

Read more: Viewing Results at Capture Points (page 525).

Viewing a Raw Multi-Variate Data Table

In some situations, it is valuable to simply view a "raw" table of all the data values (i.e., for each realization) for each output specified in a Multi-Variate result.

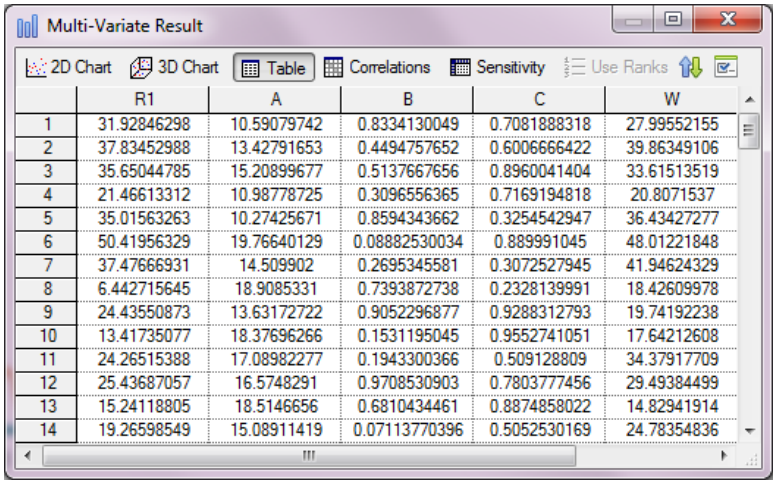
The default view for a Multi-Variate result is a 2D Scatter Plot.

Read more: Viewing a 2D Scatter Plot (page 635).

If you are viewing a 2D Scatter Plot (or a different type of Multi-variate display), you can view a Multi-Variate Data Table by pressing the **Table** button at the top of the display.

Note: When viewing a Multi-Variate Result element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

A Multi-Variate Data Table looks like this:



	R1	A	B	C	W
1	31.92846298	10.59079742	0.8334130049	0.7081888318	27.99552155
2	37.83452988	13.42791653	0.4494757652	0.6006666422	39.86349106
3	35.65044785	15.20899677	0.5137667656	0.8960041404	33.61513519
4	21.46613312	10.98778725	0.3096556365	0.7169194818	20.8071537
5	35.01563263	10.27425671	0.8594343662	0.3254542947	36.43427277
6	50.41956329	19.76640129	0.08882530034	0.889991045	48.01221848
7	37.47666931	14.509902	0.2695345581	0.3072527945	41.94624329
8	6.442715645	18.9085331	0.7393872738	0.2328139991	18.42609978
9	24.43550873	13.63172722	0.9052296877	0.9288312793	19.74192238
10	13.41735077	18.37696266	0.1531195045	0.9552741051	17.64212608
11	24.26515388	17.08982277	0.1943300366	0.509128809	34.37917709
12	25.43687057	16.5748291	0.9708530903	0.7803777456	29.49384499
13	15.24118805	18.5146656	0.6810434461	0.8874858022	14.82941914
14	19.26598549	15.08911419	0.07113770396	0.5052530169	24.78354836

Each row of the table is a different realization, while the columns show the values of the selected outputs.

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively. If the column is too narrow to display the contents of a cell, holding the cursor over the cell displays a tool-tip with the contents.

Like all result tables in GoldSim, this table can be sorted in ascending or descending order by selecting a column and pressing the **Sort** button.

Read more: Sorting Values in Result Tables (page 524).

You can copy the contents of the table to the clipboard. To do so, you must first select the entire table (by double-clicking on the empty cell in the upper left-hand corner of the table. After you do so, you can copy the table to the clipboard by **Ctrl+C**. You can subsequently paste the table into another application (such as a spreadsheet).

Note: You can control the number of significant figures displayed in tables from the Results tab of the Options dialog (accessed via Model |Options… from the main menu).

Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 524).

By default, Multi-Variate results operate on Final Values. That is, the display applies to the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

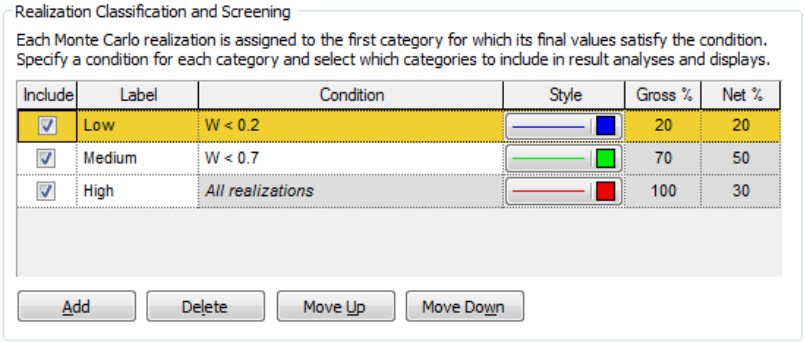
Read more: Viewing Results at Capture Points (page 525).

Using Result Classification and Screening in Multi-Variate Results

When carrying out probabilistic simulations, you may often run hundreds or thousands of realizations. In order to analyze the results, it is often quite useful to classify the realizations into *categories*. A category is simply defined by a condition relating one or more outputs in the model (e.g., those realizations in which the discount rate was above 3.5%; those realizations in which the profit exceeded $1,000,000; those realizations in which the peak concentration was between 1 mg/l and 10 mg/l).

Read more: Classifying and Screening Realizations (page 533).

Classification categories are defined at the bottom of the Monte Carlo Result Display Properties dialog:



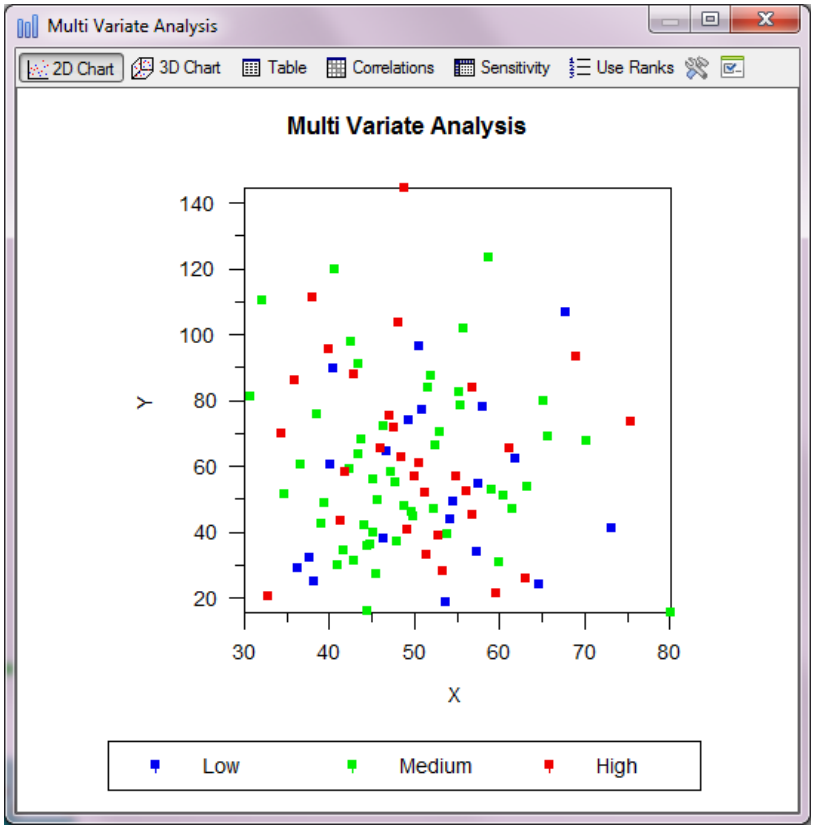
This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the **Result Options…** button), and is also accessible by pressing the **Options…** button available in the Result Properties dialog of a Multi-Variate Result element.

When viewing a Multi-Variate result, you can display a 2-D or 3-D scatter plot by selecting **2D Plot** or **3D Plot** button at the top of the display.

Read more: Viewing a 2D Scatter Plot (page 635); Viewing a 3D Scatter Plot (page 637).

In a scatter plot, each point represents a different realization. When viewing a scatter plot in a model in which categories have been defined, each category is displayed in a different color (and/or symbol).

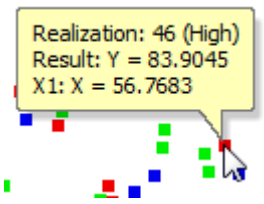
For the example categories shown above, the plot would look like this:



As can be seen, each category is presented as a different data set. Note that a legend is available for scatter plots (you may have to turn it on by right-clicking in the chart and selecting to show the legend from the context menu). The Labels specified in the Monte Carlo Result Display Properties dialog are used in the legend to label the category results.

The plot is based on the net number of realization falling into each category. In this particular example, the net number of realizations falling into each category is as follows: 18% of the realizations fall into the Low category, 36% fall into the Medium category, and 46% fall into the High category. This information is displayed in the the Monte Carlo Result Display Properties dialog after you run the simulation.

Note that if you place your cursor over any of the points, it will show a tool-tip with information regarding that particular data point:

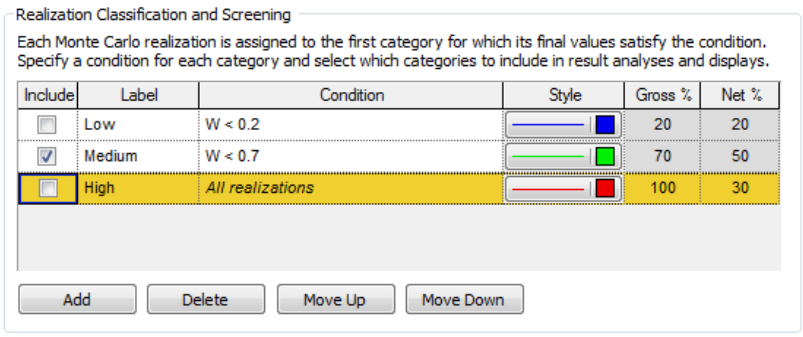


Note that for each category, you can edit the **Style** via the Monte Carlo Result Display Properties dialog. (The Style selections for each category are displayed in the Distribution Result Properties dialog, but cannot be edited there.)

Read more: Controlling the Chart Style in Multi-Variate Results (page 647).

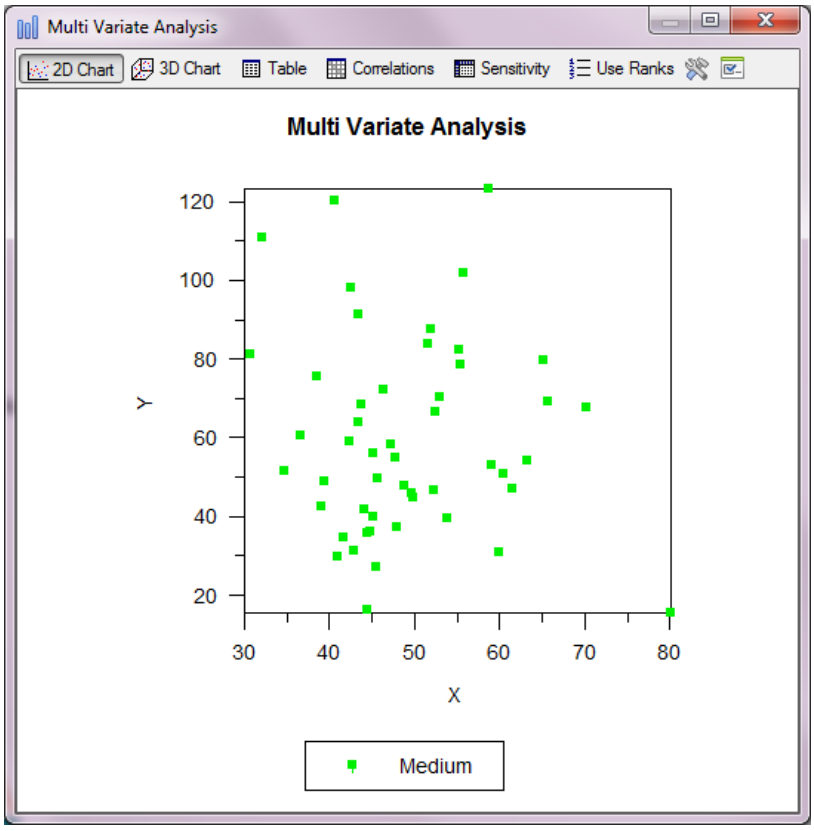
Within all Mult-Variate result displays, you can also choose to ***screen*** out one or more categories, so that the results that are shown (in charts and tables) only

include those realizations in the categories which you have chosen to include. You can screen a particular category from the results by clearing the **Include** box at the bottom of the Monte Carlo Result Display Properties dialog:



Note: You can edit the **Include** box in Edit Mode, Result Mode Scenario Mode or Run Mode.

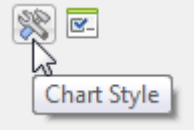
The the example above, the Low and High categories have been screened out, so when viewing a 2D Scatter Plot, only the Medium category is displayed:



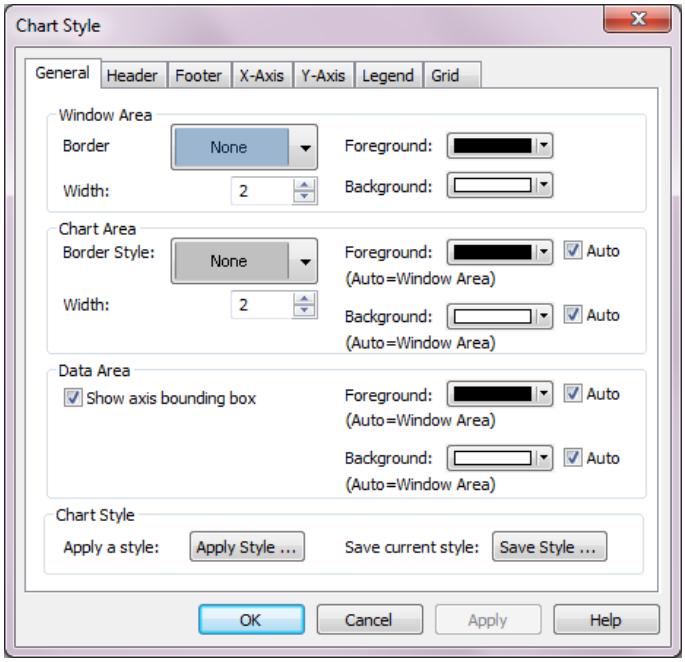
Note that if you screened categories in this way and viewed a Sensitivity Analysis, Correlation Matrix or Data Table, the analysis or display is based only on the unscreened realizations.

Controlling the Chart Style in Multi-Variate Results

GoldSim has powerful charting capabilities that allow you to customize the appearance of each chart. This includes adding headers and footers, and changing axis scales and labels for scatter plots. Most of these attributes can be edited by pressing the **Edit Chart Style** button at the top of the Multi-Variate display window:



Pressing this button (or right-clicking in a scatter plot and selecting **Edit Chart Style…**) provides access to the following dialog for editing the various chart properties:

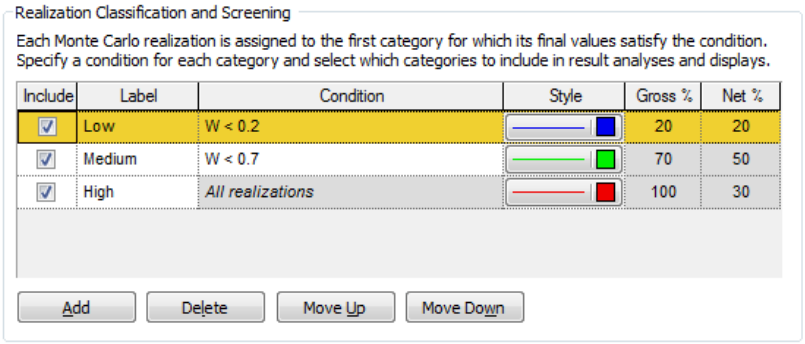


This dialog is common to all types of charts, and is discussed elsewhere.

Read more: Editing the Appearance of a Chart (page 661).

In the absence of Classification categories, the symbol used for a scatter plot is fixed and cannot be edited (a solid red aquare).

If you have defined realization categories, have run multiple realizations, then the symbol styles for the various categories can be specified in the Monte Carlo Result Display Properties dialog (most easily accessed by pressing the **Options…** button in the Result Properties dialog of a Distribution Result element). The bottom of the Monte Carlo Result Display Properties dialog looks like this:



Read more: Using Result Classification and Screening in Distribution Results (page 616).

The Style used for each category displayed in a scatter plot can be edited by clicking on the field (in the **Style** column) corresponding to each category.

Viewing Array Results

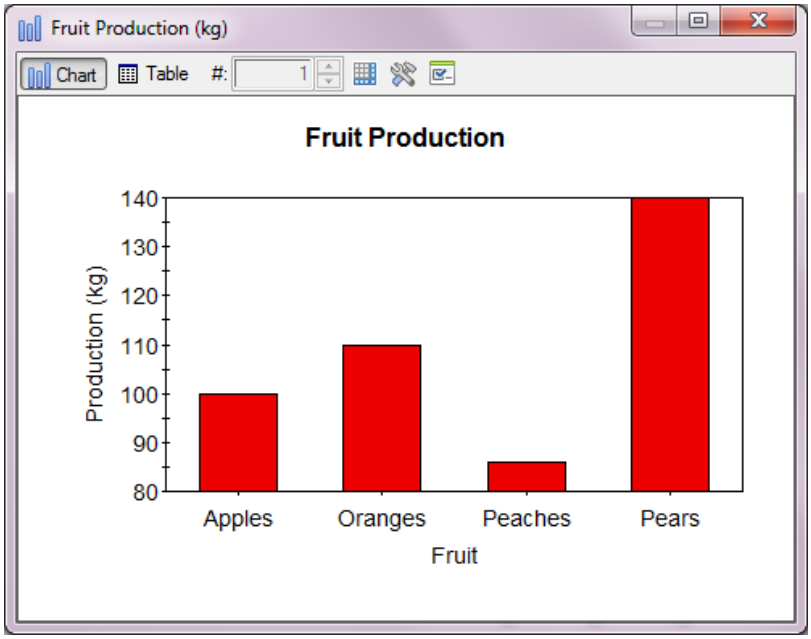
Many complex models utilize arrays (vectors and matrices).

Read more: Using Vectors and Matrices (page 726).

Array results allow you to view "snapshots" of vectors and matrices at particular points in time in both graphical and tabular form. By default, Array results display Final Values (i.e., values at the end of each realization). However, by defining Capture Points, you can display results at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 525).

If you have saved Final Values for an array output, you can display an Array result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Array Result…** from the context menu. By default, an Array Chart will be displayed:



This example shows a vector chart. The vector has four items.

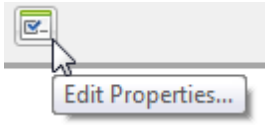
Note: When viewing an Array Result element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

Note: Array results are not available in Scenario Mode. They can only be viewed in Result Mode.

Read more: Understanding Simulation Modes (page 456).

Viewing the Properties of an Array Result

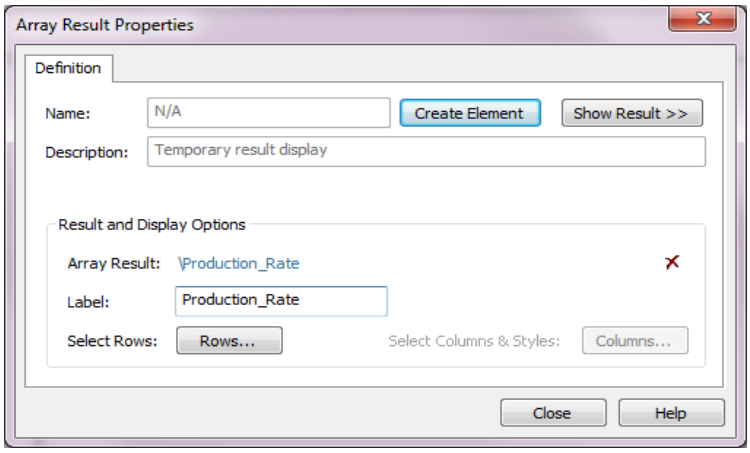
If you click on the Result Properties button at the top of the display window when viewing an Array Result chart or table, the Properties dialog for the result will be displayed:



Note that when you press this button, it does not close the result display window you were displaying. Both windows are displayed simultaneously (although you can subsequently close one or the other). When viewing results, any changes you make to the Result Properties are immediately represented in the result display.

The Result Properties dialog is always modal. That is, it cannot be minimized, and with the exception of viewing the result display (which can be displayed simultaneously and share the focus with the Properties dialog), it retains the focus while it is displayed. As a result, you must close the Result Properties dialog before you can edit any other part of your GoldSim model.

The Result Properties dialog for an interactive Array result looks like this:



At the top of the page is a button to **Create Element**. By pressing this button, a Result element is created. When you do so, the key change that will be noticeable is that the **Name** will now be editable. The **Create Element** button becomes an **Appearance** button (which is used to modify the appearance of the element itself).

Read more: Creating and Using Result Elements (page 526).

The **Show Result >>** button opens a result display window (while keeping the Properties dialog simultaneously open.

Read more: Viewing a Vector Chart (page 651); Viewing a Matrix Chart (page 653); Viewing an Array Table (page 656).

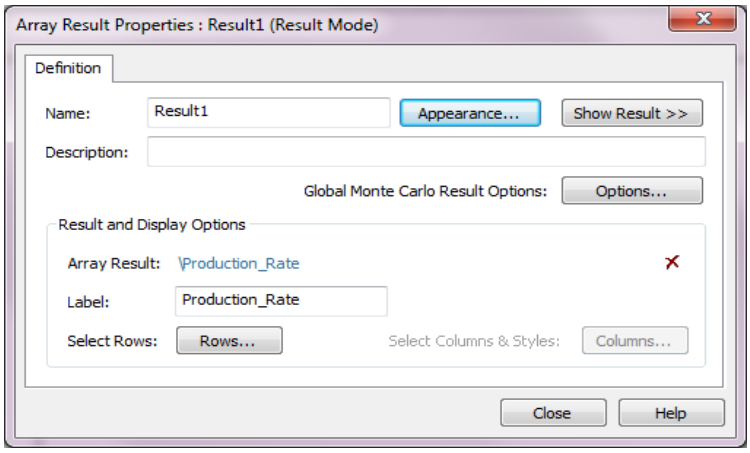
The output being displayed by the Array result is listed directly to the right of the **Array Result** text. If you click on the ouput, it will open a browser, allowing you to select a different array (you must select an array; GoldSim will not allow you to select a scalar output). If you click on the **Remove** button (the red script X just above the **Columns…** button), it will remove the selected output (and replace it with "<Click to select>". You would then need to click this to select another array.

The **Label** defaults to the name of the result being displayed. However, you can edit this (to make it more readable). It is displayed on the vertical axis of charts.

The **Rows…** and **Columns…** buttons provide access to a dialog for selecting which items to show in the display (and in the case of matrices, the **Columns…** button also provides access to a dialog for selecting the style of the bars shown in the chart). The **Columns…** button is only available for matrices.

Read more: Controlling the Chart Style in Array Results (page 659).

The Properties dialog for an Array Result element has several differences:



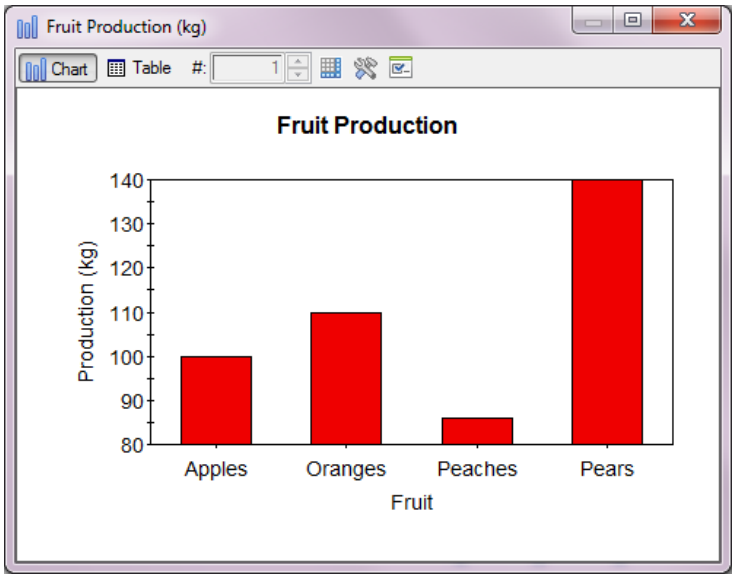
- The **Create Element** button is replaced by an **Appearance** button (which is used to modify the appearance of the element itself).
- The **Name** (and **Description**) are editable (since this is an element).
- An **Options…** button is available that provides access to the Monte Carlo Result Display Properties.

Read more: Controlling Monte Carlo Result Options (page 441).

Viewing a Vector Chart

If you have saved Final Values for a vector output, you can display the final values of the vector by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Array Result…** from the context menu.

By default, a Vector Chart will be displayed:



If you are viewing an Array Table, you can view a Vector Chart by pressing the **Chart** button at the top of the display.

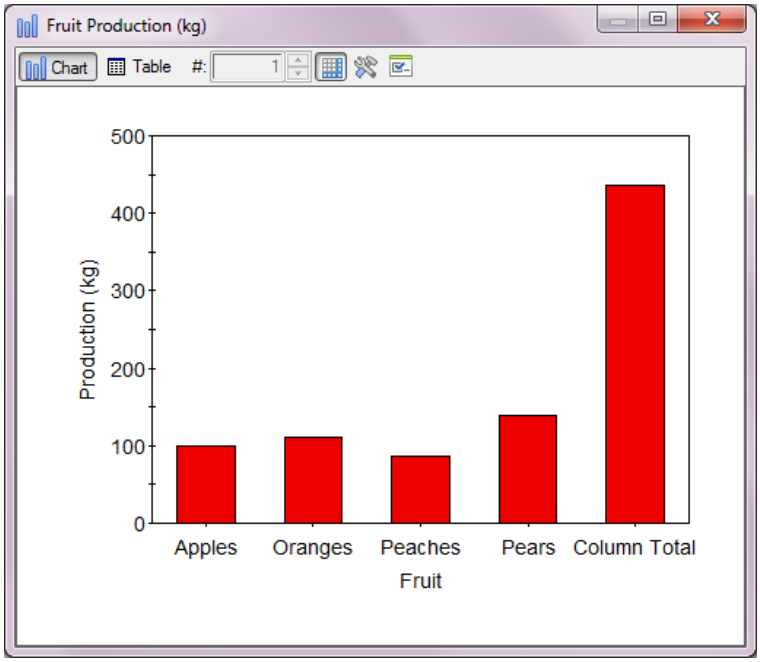
Note: When viewing an ArrayResult element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

By pressing the **Rows…** button from the Result Properties dialog for the Array result, you can select which items in the array are to be displayed (by default, all are displayed).

Read more: Viewing the Properties of an Array Result (page 649); Controlling the Chart Style in Array Results (page 659).

A row/column totals button is available at the top of the display window:

Pressing this button displays an additional bar with the total of al the row values:



Vector charts have several hot-keys which can be used to modify the plot:

Keys	Action
Ctrl+S	Switches between a bar chart and line chart.
Ctrl+W	Increases the width of the bars in a bar chart.
Ctrl-Shft+W	Decreases the width of the bars in a bar chart.

The following six hot-keys allow you to rotate and view the chart in three dimensions:

Key	Action
F6	Increases the depth (into the page) of the bars.
Shft+F6	Decreases the depth (into the page)of the bars.
F7	Increases the vertical inclination from which the chart is viewed.
Shft+F7	Decreases the vertical inclination from which the chart is viewed.
F8	Rotates the chart to the left.

Key	Action
Shft+F8	Rotates the chart to the right.

You can copy the chart to the clipboard by right-clicking in the chart, and selecting **Copy** from the context menu. Alternatively, you can press **Ctrl+C** while viewing the chart. To export a chart, right-click in the chart, and select **Export…** from the context menu.

Read more: Copying a Chart or Table (page 691); Exporting a Chart (page 691).

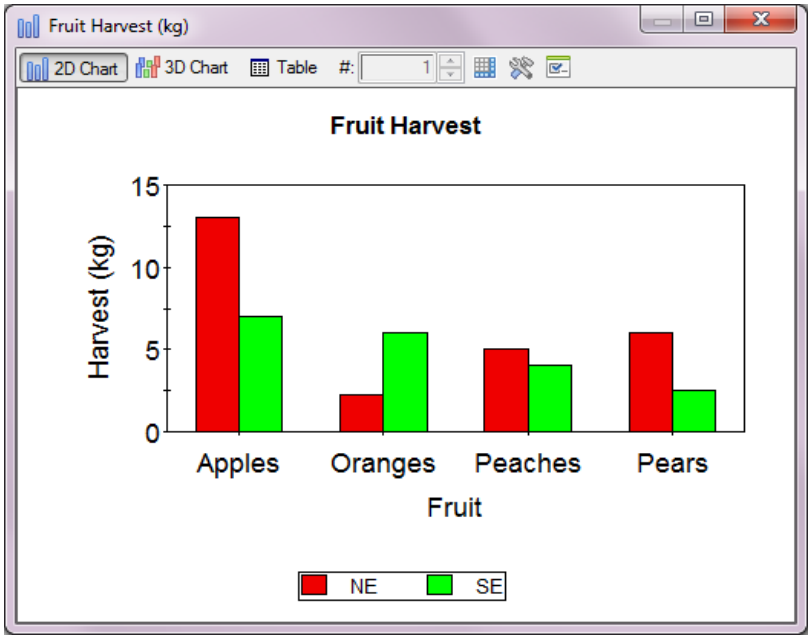
By default, Array results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 525).

Viewing a Matrix Chart

If you have saved Final Values for a matix output, you can display the final values of the matrix by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Array Result…** from the context menu.

By default, a Matrix Chart will be displayed:



The row items are plotted along the X axis, with multiple bars for each column item (which are identified in a legend, which you may have to turn on by right-clicking in the chart and selecting to show the legend from the context menu).

If you are viewing an Array Table, you can view an Array Chart by pressing the **2D Chart** button at the top of the display.

Note: When viewing an ArrayResult element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

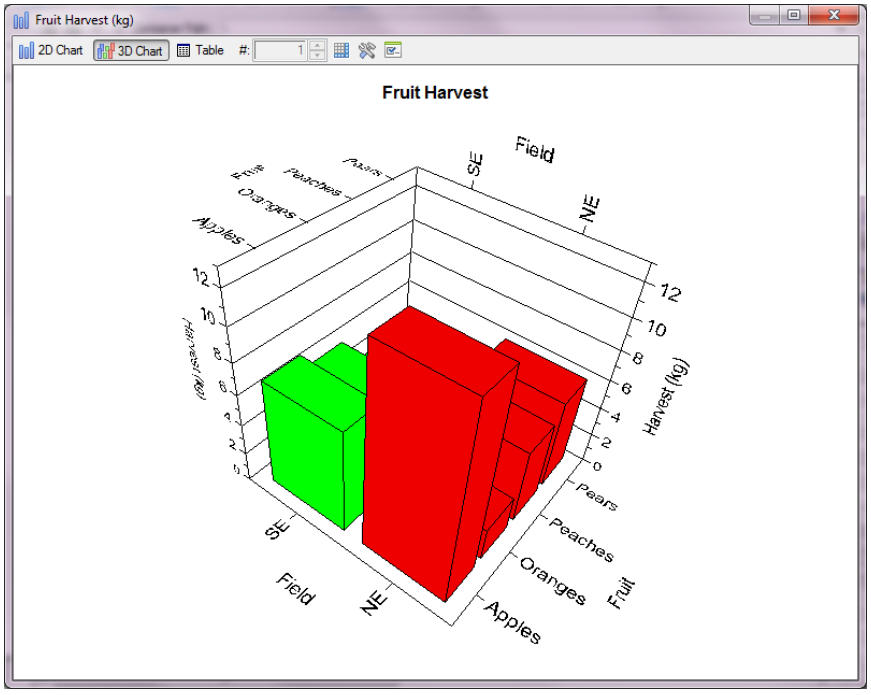
Matrix charts have several hot-keys which can be used to modify the plot:

Keys	Action
Ctrl+S	Toggles between a bar chart and line chart.
Ctrl+W	increases the width of the bars in a bar chart.
Ctrl-Shft+W	decreases the width of the bars in a bar chart.

The following six hot-keys allow you to rotate and view the chart in three dimensions:

Keys	Action
F6	increases the depth of the bars.
Shft+F6	decreases the depth of the bars.
F7	increases the vertical inclination from which the chart is viewed.
Shft+F7	decreases the vertical inclination from which the chart is viewed.
F8	rotates the chart to the left.
Shft+F8	rotates the chart to the right.

Pressing the **3D Chart** button displays the chart in three dimensions:



Note: In order for the labels to be legible for a 3D Chart, you will likely have to expand the window. The axes labels scale with the size of the window in 3D charts.

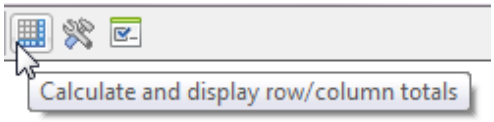
You can rotate the three dimensional image using by holding down the left mouse button and dragging within the chart. The following hot-keys can be used to manipulate the 3D Chart:

Keys	Actions
F6	increases the depth (into the page) of the bars.
Shft+F6	decreases the depth (into the page) of the bars.

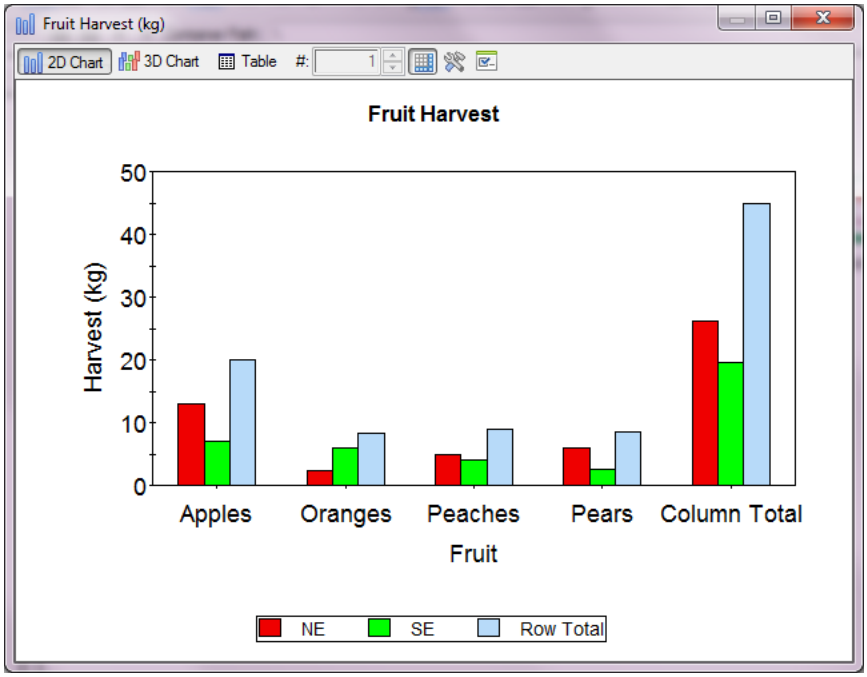
By pressing the **Rows…** and/or **Columns…** buttons from the Result Properties dialog for the array result, you can select which items in the array are to be displayed (by default, all are displayed). You can also change the colors for the column bars.

Read more: Viewing the Properties of an Array Result (page 649); Controlling the Chart Style in Array Results (page 659).

A row/column totals button is available at the top of the display window:



Pressing this button displays the row and column totals:



You can copy the chart to the clipboard by right-clicking in the chart, and selecting **Copy** from the context menu. Alternatively, you can press **Ctrl+C** while viewing the chart. To export a chart, right-click in the chart, and select **Export…** from the context menu.

Read more: Copying a Chart or Table (page 691); Exporting a Chart (page 691).

By default, Array results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 525).

Plotting Condition Arrays

Condition outputs are either True or False and are typically used as state variables or flags in a simulation. You can create arrays of conditions, and, in some situations, you may want to plot these arrays. To facilitate this, when plotting an array of a condition, GoldSim plots True as 1 and False as 0.

Read more: Understanding Output Attributes (page 93).

Viewing an Array Table

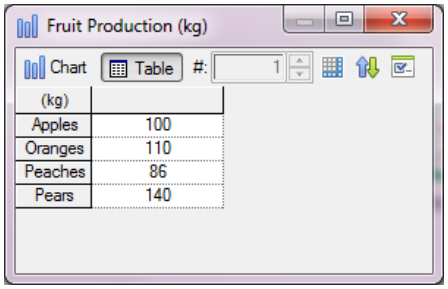
The default view for an Array result is a 2D Chart.

Read more: Viewing a Vector Chart (page 651); Viewing a Matrix Chart (page 653).

In some cases, however, you may want to view a table of values.If you are viewing a 2D Chart (or 3D Chart), you can view an Array Table by pressing the **Table** button at the top of the display.

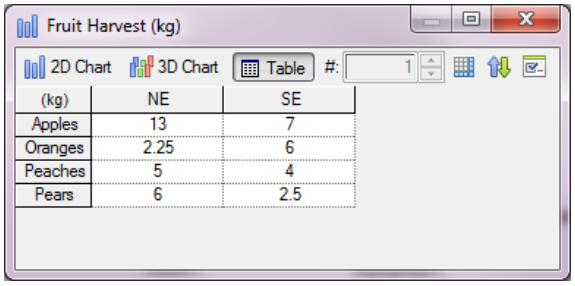
Note: When viewing an ArrayResult element, the element "remembers"the last type of view that was displayed, and displays that view when you double-click on it.

An Array Table for a vector looks like this:



The first column shows the item names for the vector. The units for the vector are in the first cell in the upper left-hand corner of the table.

An Array Table for a matrix looks like this:



The units for the vector are in the first cell in the upper left-hand corner of the table. The first row shows the item names for the columns. The first column shows the item names for the rows.

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively.

Note: Conditions are displayed in tables as either 1/0, true/false, True/False, TRUE/FALSE, on/off, or High/Low. You select which of these pairs to use on the Results tab of the Options dialog (accessed via **Model |Options…** from the main menu). The default setting is true/false.

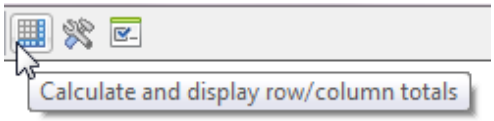
Note: You can control the number of significant figures displayed in tables from the Results tab of the Options dialog (accessed via Model |Options… from the main menu).

Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 524).

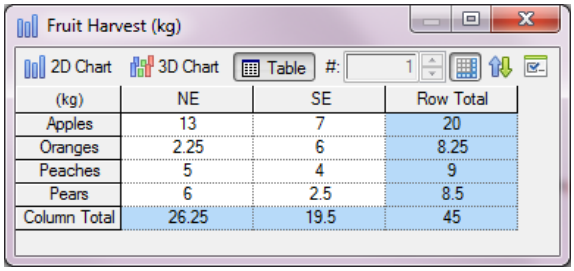
By pressing the **Rows…** and/or **Columns…** buttons from the Result Properties dialog for the Array result, you can select which items in the array are to be displayed in the table (by default, all are displayed).

Read more: Viewing the Properties of an Array Result (page 649).

A row/column totals button is available at the top of the display window:



Pressing this button displays the column and row (for matrices) totals in the table:



(kg)	NE	SE	Row Total
Apples	13	7	20
Oranges	2.25	6	8.25
Peaches	5	4	9
Pears	6	2.5	8.5
Column Total	26.25	19.5	45

You can copy the contents of an array table to the clipboard by pressing **Ctrl+C**. You must first select the cells you wish to copy. You can do so by placing your cursor in one cell and dragging to another location. You can select the entire table by pressing the cell in the upper left-hand corner of the table. Cells that are not adjacent can be selected using **Ctrl** key when selecting. Selected items will be highlighted in black. You can subsequently paste the table into another application (such as a spreadsheet).

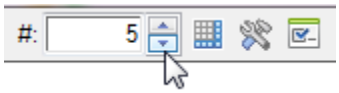
Read more: Selecting Items and Copying Values in Result Tables (page 523).

By default, Array results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 525).

Viewing Multiple Realizations of Array Results

If you have run (and saved) multiple realizations, a realization control is added to the top of the display window for an Array result:.



You can type in a specific realization (and press Enter), move upward or downward through the available realizations using the spin control, or use the **Ctrl-Up** and **Ctrl-Down** keys to view the previous or next available realization, respectively.

Note: You can only view those realizations which are available (i.e., those which have been saved and have not been screened out).

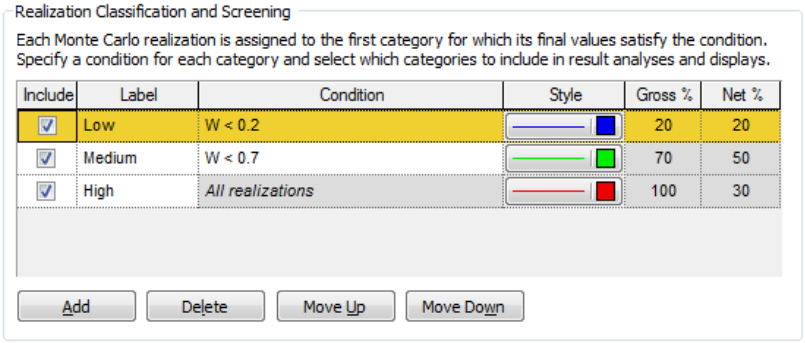
Read more: Using Result Screening in Array Results (page 658).

Using Result Screening in Array Results

When carrying out probabilistic simulations, you may often run hundreds or thousands of realizations. In order to analyze the results, it is often quite useful to screen out some of the realizations by classifying the realizations into *categories*. A category is simply defined by a condition relating one or more outputs in the model (e.g., those realizations in which the discount rate was above 3.5%; those realizations in which the profit exceeded $1,000,000; those realizations in which the peak concentration was between 1 mg/l and 10 mg/l).

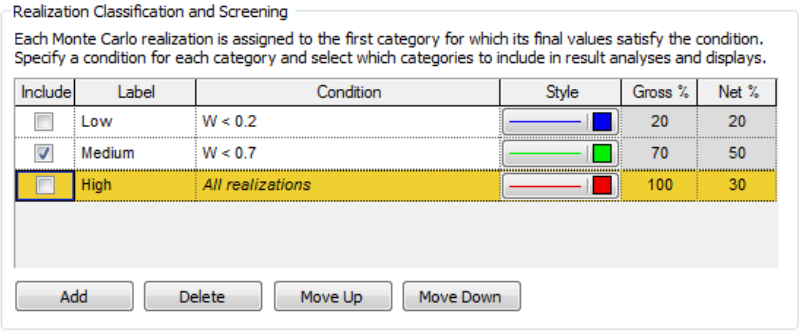
Read more: Classifying and Screening Realizations (page 533).

Classification categories are defined at the bottom of the Monte Carlo Result Display Properties dialog:



This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the **Result Options…** button), and is also accessible by pressing the **Options…** button available in the Result Properties dialog of an Array Result element.

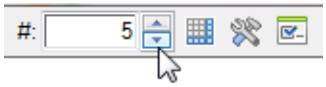
You can screen a particular category from the results by clearing the **Include** box at the bottom of the Monte Carlo Result Display Properties dialog:



Only include those realizations in the categories which you have chosen to include can then be displayed in charts and tables.

Note: You can edit the **Include** box in Edit Mode, Result Mode Scenario Mode or Run Mode.

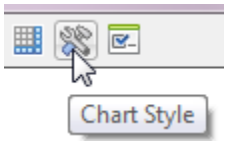
In the example above, the Low and High categories have been screened out, so when displaying results, only those realizations that fall into the Medium category would be available in the realizations spin control at the top of the display window:



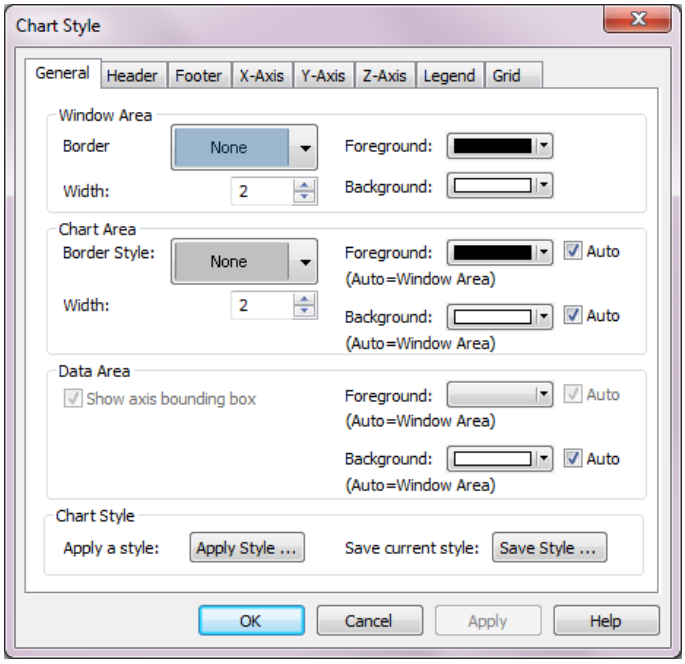
Read more: Viewing Multiple Realizations of Array Results (page 658).

Controlling the Chart Style in Array Results

GoldSim has powerful charting capabilities that allow you to customize the appearance of each chart. This includes adding headers and footers, and changing axis scales and labels for scatter plots. Most of these attributes can be edited by pressing the **Edit Chart Style** button at the top of the Multi-Variate display window:



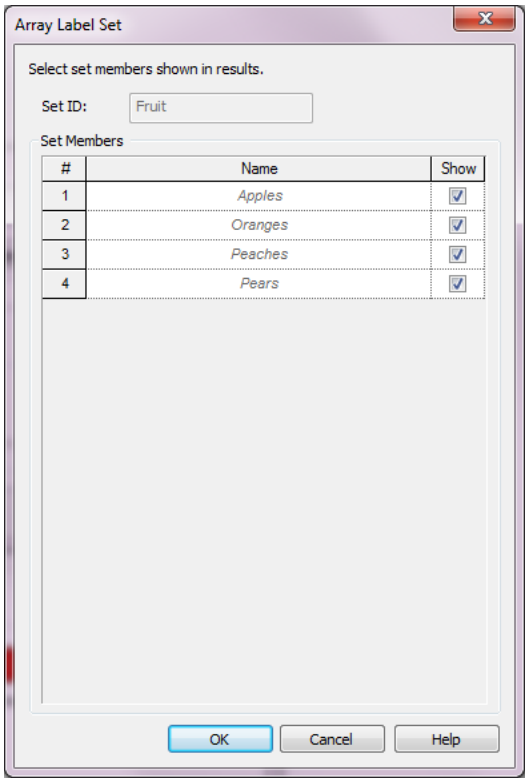
Pressing this button (or right-clicking in a chart and selecting **Edit Chart Style…**) provides access to the following dialog for editing the various chart properties:



This dialog is common to all types of charts, and is discussed elsewhere.

Read more: Editing the Appearance of a Chart (page 661).

For Vector Charts, the bar chart color is is fixed and cannot be edited (solid red). You can, however, modify which items are actually included in the chart. Pressing the **Rows…** button from the Result Properties dialog for the Array result displays the following dialog:

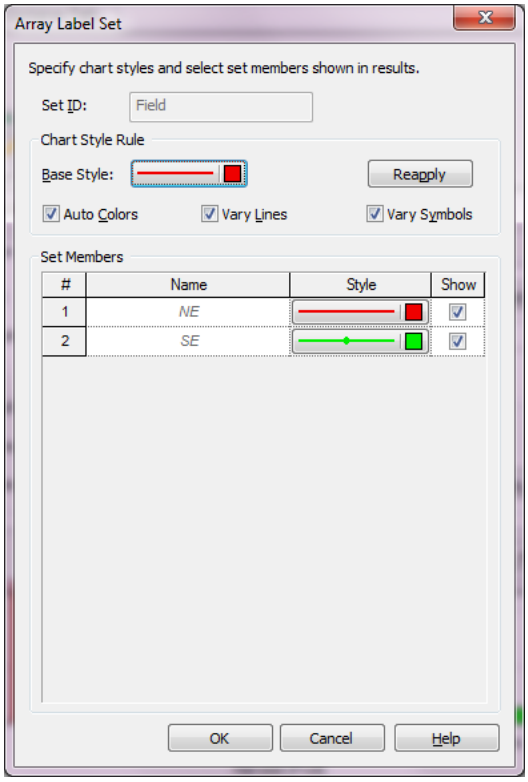


Read more: Viewing the Properties of an Array Result (page 649).

Note: This dialog can also be accessed from the main GoldSim menu (by selecting **Model | Array Labels…** and then selecting the appropriate Array Label Set). When you access the dialog in this manner, the Style for each item is also displayed, but changing these has no impact on Vector Charts, since the Style is fixed.

By default, the **Show** box is checked for all items. If you clear this box, the row item will not be displayed in charts (or tables).

For Matrix Charts, the bar chart color for each column can be specified by pressing the **Columns…** button from the Result Properties dialog for the Array result. The following dialog will be displayed:



Note: This dialog can also be accessed from the main GoldSim menu (by selecting **Model | Array Labels…** and then selecting the appropriate Array Label Set).

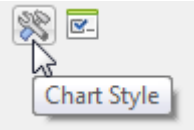
The **Style** column controls the style for the bars for the columns in the Matrix Chart. You can also modify which column items are actually included in the chart. By default, the **Show** box is checked for all items. If you clear this box, the column item will not be displayed in charts (or tables).

Editing the Appearance of a Chart

Whenever you first view a result in GoldSim, it takes on a set of characteristics (e.g., fonts, axis titles, header) defined by a default chart style for that particular type of chart. In some cases, these default chart styles will be sufficient.

In other cases, however, you will want to customize the appearance of the chart. GoldSim allows you to do so. In addition, after customizing a set of properties for a particular type of chart, you can save this set as a new style which you can then selectively apply to subsequent charts (or set as the default for all subsequent charts of that type).

Selecting the **Chart Style…** button in any chart (or right-clicking on any chart and selecting **Edit Chart Style** from the context menu) displays the Chart Style dialog for editing the appearance of the chart:



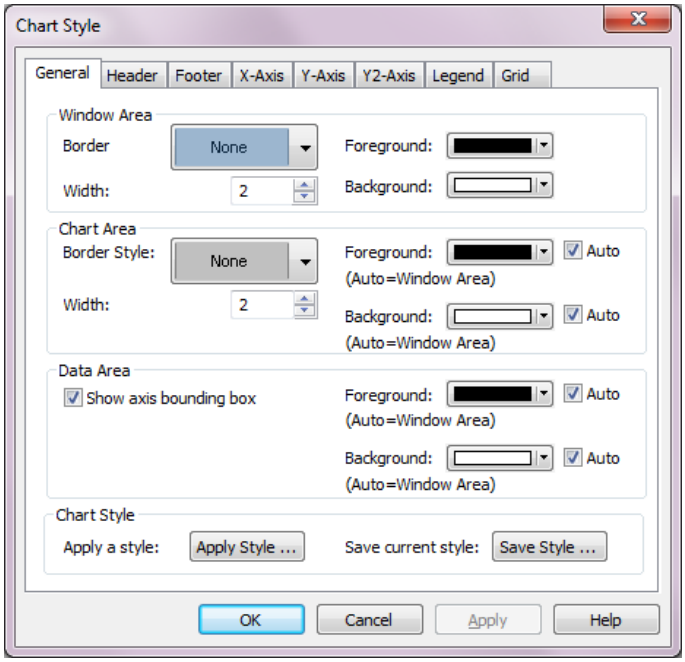
This dialog allows you to edit the appearance of the chart (e.g., by modifying headers, footers, axis labels and scales, legends, etc.). To make it easier to modify and customize your charts, the same tabbed dialog is used for all chart types in GoldSim; some of the tabs and/or options, however, are hidden for those charts for which they are not applicable.

Note: When this dialog is displayed, the Chart display stays active. If you modify some part of the chart's appearance using the dialog and press the Apply button, the change is applied to the Chart. This allows you to view changes without closing the dialog.

The Chart Style General Tab

The dialog has a number of tabs, which are described in the following sections.

The General tab is shown by default when you first open the Chart Style dialog.



The General tab serves two purposes:

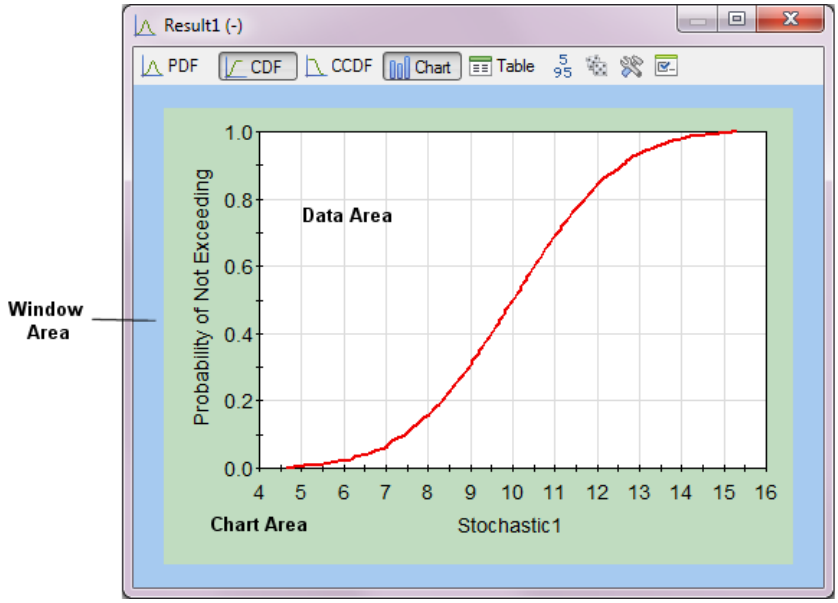
- The upper portion of the tab allows you to specify the colors and border styles for the chart; and

- The lower portion of the tab allows you to apply a specific chart style (a common group of appearance properties than can be applied to the entire chart), and (if changes have been made) allows you to overwrite the current chart style or save the current set of properties as a new chart style.

Read more: Creating and Using Chart Styles (page 671).

The upper portion of the **General** tab of the dialog is described below.

Colors are defined in a hierarchical manner for a chart by Window Area, Chart Area, and Data Area. The Window Area encompasses the Chart Area, and the Chart Area encompasses the Data Area, as illustrated below:



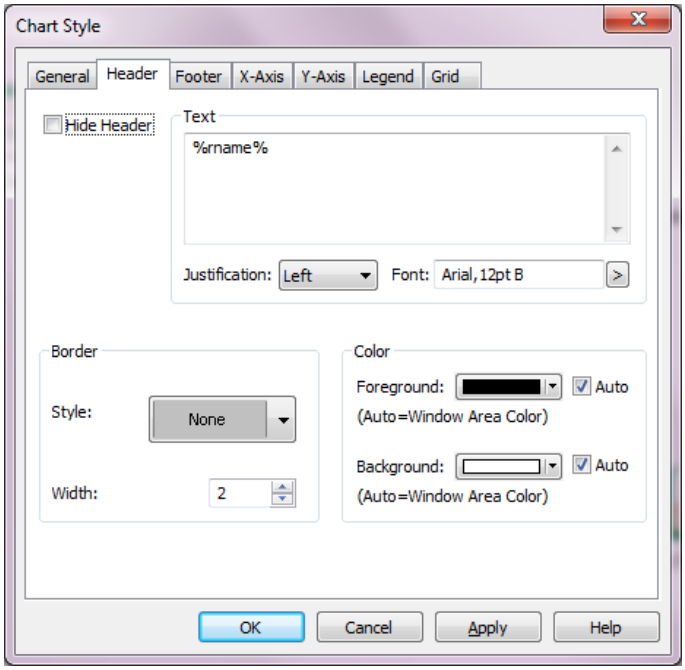
By default, the Chart Area defaults to the same colors as the Window Area and the Data Area defaults to the same colors as the Chart Area.

The Foreground Color controls the text or lines that appear in the chart. The Window Area Foreground Color controls the header, the footer and the legend. The Chart Area Foreground Color controls the axes, the axis labels and the grid.

You can place a border around the Window Area and/or the Chart Area of a specified style and width. You can also choose to display a bounding box around the Data Area by checking **Show axis bounding box**.

The Chart Style Header and Footer Tabs

The Header and Footer tabs are used to specify a header and footer for the chart, and are identical in appearance.



If **Hide Header** (or **Hide Footer**) is checked, the header or footer is hidden. You can also hide or show the header or footer via the chart context menu (under View) or via the header/footer context menu.

You enter **Text** for the header or footer, and determine the **Justification** (if it consists of multiple lines), and **Font**.

You can place a border around the header or footer, and specify its **Style** and **Width** (in terms of pixels). You can also specify the **Foreground** and **Background** color for the header or footer (or default the colors so they use that of the Window Area, as specified in the General tab).

Note: You can access the Header or Footer tab of the chart style dialog directly by selecting **Edit Header…** or **Edit Footer…** from the header/footer context menu.

Note: If you right-click within the **Text** edit field, a context menu providing a list of keywords will be provided. Keywords allow you to automatically insert text that is determined automatically by the context of the result (e.g., the keyword %Rname% inserts the Result element name). This allows you to use a single chart style for multiple results.

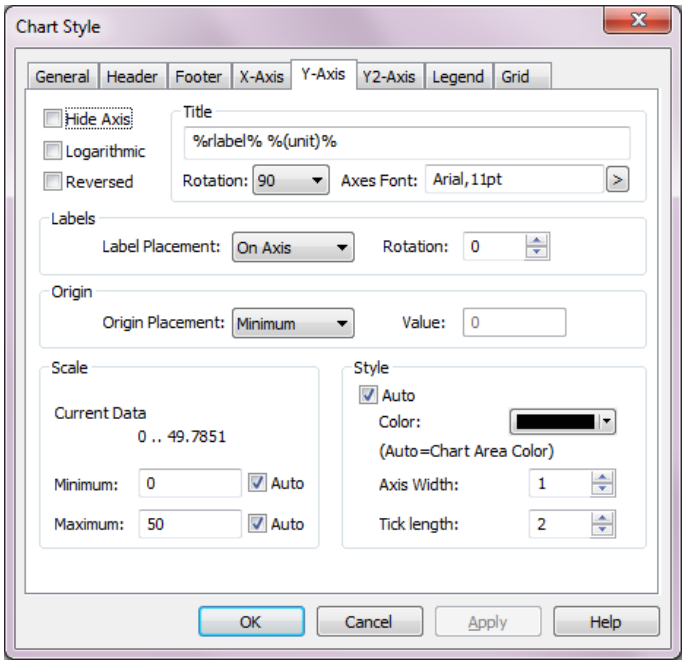
Read more: Using Keywords in Styles (page 676).

The Chart Style Axis Tabs

Depending on the type of chart and the number of outputs being plotted, up to three axis tabs may be present in the Chart Style dialog (there are always at least two):

Chart Type	X-axis	Y-axis	Y2-axis	Z-axis
Time History	X	X	X	
Distribution	X	X		
Scatter Plot	X	X		X
Array	X	X		X

The X, Y and Y2 axis tabs have the same appearance (with one minor exception noted below).



At the top of the dialog, GoldSim indicates what the particular axis represents, along with its units (if applicable).

The various fields in this tab are described below:

Title: This is the axis title. If you right-click within the **Title** edit field, a context menu providing a list of *keywords* will be provided. Keywords allow you to automatically insert text that is determined automatically by the context of the result (e.g., the keywords %rlabel% %(unit)% inserts the result label and the display units for the output associate with the axis). This allows you to use a single chart style for multiple results.

Read more: Using Keywords in Styles (page 676).

Title Rotation: This is only available for the Y and Y2 axes. There are three options: 90 (the default), in which the title is rotated and read from bottom to top; 270, in which the title is rotated and read from top to bottom; and None, in which the title is not rotated (i.e., is horizontal), and is placed at the top of the axis.

Axis Font: This provides access to a dialog for specifying the font for the axis. Note that this font is applied to all axes (i.e., all axes must use the same font).

Hide Axis: If this box is checked the entire axis (and all annotation) is hidden.

Logarithmic: If this box is checked, the axis is plotted logarithmically (rather than linearly). Note that if the data set you are plotting contains numbers which are less than or equal to zero, these points cannot be plotted on a logarithmic axis. Therefore, if you make the axis logarithmic in such a situation, GoldSim will omit these points from the plot (i.e., they will appear as "holes" or "gaps" in the plot).

Note: Bar charts cannot be plotted on a logarithmic axis. For example, when plotting a distribution chart as a PDF, the X-axis cannot be logarithmic. If you check the **Logarithmic** box, it will be ignored. In addition, 2-D and 3-D scatter plots cannot be plotted using logarithmic axes. In these cases, the **Logarithmic** box is grayed out.

Reversed: If this box is checked, the axis numbering is reversed (e.g., going from high values to low values rather than from low values to high values).

Label Placement: This is only available for the X, Y and Y2 axes, and determines where the axis labels are placed relative to the other (orthogonal) axis. There are three options: On Axis (the default), in which the labels are placed alongside the axis, Minimum (in which they are placed at the minimum value of the other axis), or Maximum (in which they are placed at the maximum value of the other axis).

Label Rotation: This determines the angle that the axis numbers are rotated. The default is 0.

Origin Placement: This is only available for the X, Y and Y2 axes, and determines where the axis intersects the orthogonal axis (e.g., where the X-axis intersects Y-axis). There are three options: Minimum (the default), in which it intersects at the minimum value of the other axis; Maximum, in which it intersects at the maximum value of the other axis; or At Value, in which it intersects at the **Value** on the other axis indicated directly to the right.

Note: You cannot select label or origin placement for the Y axes for a time history chart with two vertical (Y and Y2) axes.

Scale Minimum: This is the minimum value (lower bound) on the axis. If **Auto** is checked, GoldSim will select the value for you (based on the range).

Scale Maximum: This is the maximum value (upper bound) on the axis. If **Auto** is checked, GoldSim will select the value for you (based on the range).

Note: If the axis being plotted is a date, the scale minimum and maximum will display date and time controls so you can specifically select a date/time range.

Color: This is the color of the axis and all of the annotation. If **Auto** is checked, the foreground color specified for the Chart Area (in the **General** tab) will be used.

Axis Width: This is the width of the axis lines, in pixels

Tick Length: This is the tick length, in pixels.

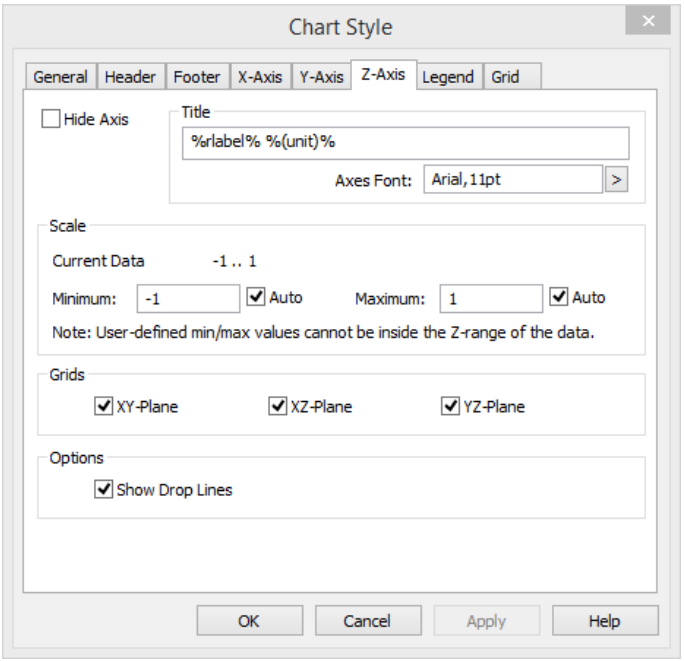
Note: One key attribute for charts is how values on axes are displayed. You cannot control the number of significant figures displayed (this is automatically determined). You can, however, control under what circumstances scientific notation is used via the **Results** tab of the Options dialog (accessed via **Model |Options…** from the main menu).

Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 524).

Note: When viewing a 3D scatter plot, some of the axis options are fixed by GoldSim and will automatically be grayed out in the X-Axis and Y-Axis tabs.

Specifying the Axis Appearance for 3D Charts

The Z-axis tab of the Chart Style dialog box is only available when viewing 3D scatter plots and 3D array plots and contains a subset of the options available on the other axes.



This is because the nature of a 3D plot is such that other than the title, the scale, the font and the font size, the appearance of the axis labels in a 3D plot are fixed by GoldSim and cannot be edited.

The various fields in this tab are described below:

Title: This is the axis title. If you right-click within the **Title** edit field, a context menu providing a list of *keywords* will be provided. Keywords allow you to automatically insert text that is determined automatically by the context of the **result** (e.g., the keywords %rlabel% %(unit)% inserts the result label and the display units for the output associate with the axis). This allows you to use a single chart style for multiple results.

Read more: Using Keywords in Styles (page 676).

Axis Font: This provides access to a dialog for specifying the font for the axis. Note that this font is applied to all axes (i.e., all axes must use the same font).

Hide Axis: If this box is checked the entire axis (and all annotation) is hidden.

Scale Minimum: This is the minimum value (lower bound) on the axis. If **Default** is checked, GoldSim will select the value for you (based on the range).

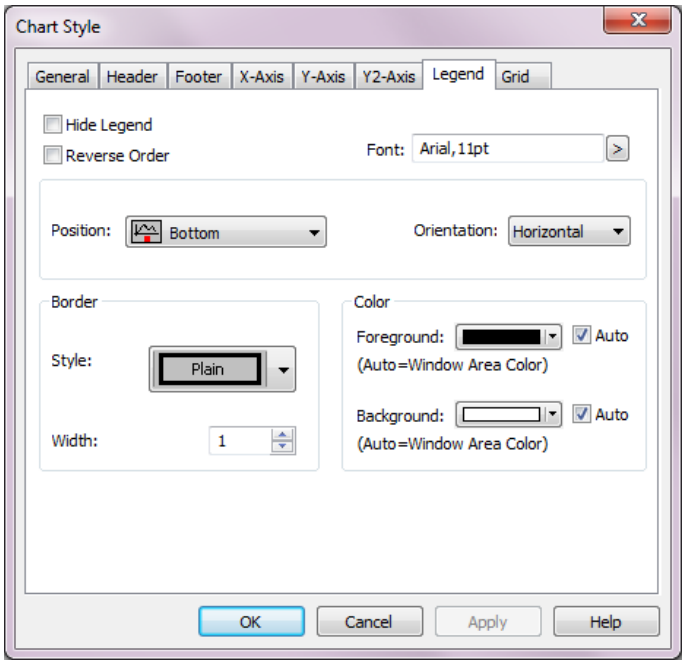
Scale Maximum: This is the maximum value (upper bound) on the axis. If **Default** is checked, GoldSim will select the value for you (based on the range).

Grids (XY-Plane, XZ-Plane, YZ-Plane): This determines if grids are shown for the specified planes. If checked (and applicable to the plot), the grids are shown.

Show Drop Lines: If this box is checked, GoldSim will draw a line from the data point to the XY-plane for scatter plots. It does not apply to array charts.

The Chart Style Legend Tab

The Legend tab of the Chart Style dialog is used to specify the legend for the chart, and is only available for those charts for which a legend is applicable.



If **Hide Legend** is checked, the legend is hidden. You can also hide or show the legend via the chart context menu (under View) or via the legend context menu.

You enter the **Title** for the legend, and determine its **Font**. If the **Reverse Order** box is checked, the items of the legend are listed in reverse order.

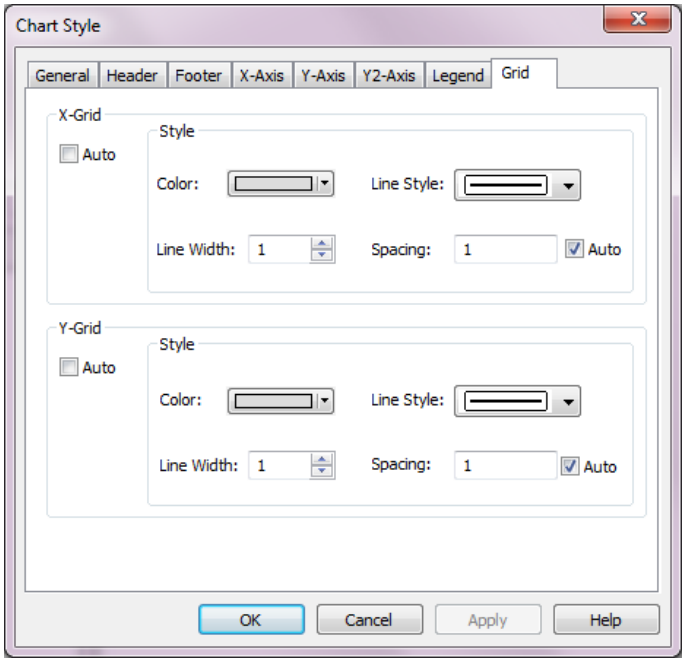
The **Position** field allows you to place the legend at various locations around the chart (Top, Bottom, Right, Left, etc.). The **Orientation** of the items in the legend can be specified as being vertical or horizontal.

You can place a border around the legend, and specify its **Style** and **Width** (in terms of pixels). You can also specify the **Foreground** and **Background** color

for the legend (or default the colors by selecting **Auto** so they use that of the Window Area, as specified in the **General** tab).

The Chart Style Grid Tab

The Grid tab of the Chart Style dialog is used to specify the grid for the chart.



Note that the grid lines are specified for each axis (and are perpendicular to the axis).

You can specify the **Line Style** and **Line Width** (in terms of pixels). You can also specify the **Color** for the grid lines.

You specify the **Spacing** of the grid lines in terms of the actual units for the data assigned to the axis.

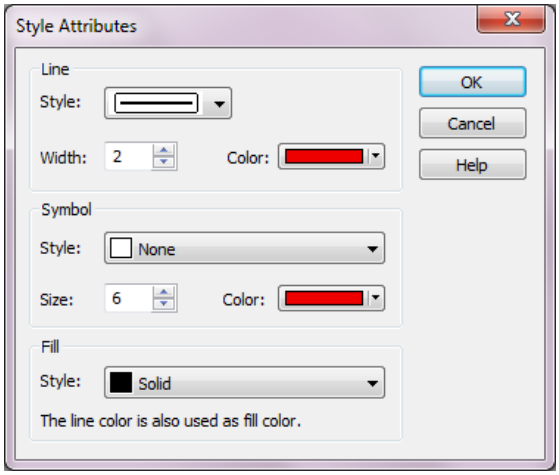
If **Auto** is checked, GoldSim automatically computes an appropriate spacing based on the data range, defaults the color so it uses the Chart Area foreground color (as specified in the **General** tab), and resets the **Line Style** and **Line Width** to default settings.

Editing Data Styles

Unlike the other attributes of a chart (e.g., axes labels, headers, legends), the manner in which the data itself is shown in the chart is not a property of the chart (and hence is not edited in the Chart Style dialog), but is determined by the type of result and the context in which it is being plotted (e.g., is the result an array, are scenarios being viewed, have classification categories been defined).

As such, Data Styles can be specified in multiple places within GoldSim and the style that is used for a particular chart is a function of the context for the display (discussed below).

In all cases, however, the Data Style dialog for a particular data set looks like this:



For each data set, you can select the **Line Style** (e.g., solid, dashed, etc.), the **Line Width** (in pixels) and the **Line Color**. You can also choose to plot a symbol at the data points (the default is no symbol is shown), and select a **Symbol Style** (e.g., square, solid circle, etc.), **Symbol Size** (in pixels), and **Symbol Color**. For bar charts, you can also choose the **Fill Style**. The selected line color is used as the fill color.

There are several different places from where this dialog can be accessed. It is easiest to summarize this by examining each type of result separately.

Time History Results

- In most situations, the Data Style is specified directly in the Result Properties dialog.

  Read more: Viewing the Properties of a Time History Result (page 537).

- If you are plotting an array, the Data Style for the different array items is defined in the Array Label Set dalog.

  Read more: Viewing Time Histories for Array Outputs (page 547).

- If you have run multiple realizations and you are displaying **Probabilities**, the Data Style for the display is specified at the top of the Monte Carlo Resilt Display Propertites dialog (in the section labled "History Statistics"). This dialog is accessed by pressing the **Options…** button in the Result Properties dialog of a Time History Result element.

  Read more: Viewing Probability Histories for Multiple Realizations (page 555).

- If you have run multiple realizations; 2) you have defined more than one category; and 3) you are displaying **All Realizations**, GoldSim will label the curves in the Time History Chart based on the categories you have defined. The Data Style for categories is defined at the bottom of the Monte Carlo Resilt Display Propertites dialog (in the section labled "Realization Calssification and Screening"). This dialog is accessed by pressing the **Options…** button in the Result Properties dialog of a Time History Result element.

  Read more: Using Result Classification and Screening in Time History Results (page 571).

- When a model is in Scenario Mode, the Data Style for the different scenarios is controlled by the Scenario Manager dialog (accessed from the main menu (**Run | Scenario Manager…**) or by pressing **F7**).

  Read more: Viewing Scenario Results in Time History Result Elements (page 578).

Distribution Results

- In most situations, the Data Style is specified directly in the Result Properties dialog.

  Read more: Viewing the Properties of a Distribution Result (page 597).

- If you have run multiple realizations; 2) you have defined more than one category; and 3) the Distribution Result has only a single result defined, GoldSim will label the curves in the Distribution Chart based on the categories you have defined. The Data Style for categories is defined at the bottom of the Monte Carlo Resilt Display Propertites dialog (in the section labled "Realization Calssification and Screening"). This dialog is accessed by pressing the **Options…** button in the Result Properties dialog of a Time History Result element.

  Read more: Using Result Classification and Screening in Distribution Results (page 616).

- When a model is in Scenario Mode, the Data Style for the different scenarios is controlled by the Scenario Manager dialog (accessed from the main menu (**Run | Scenario Manager…**) or by pressing **F7**).

  Read more: Viewing Scenario Results in Distribution Result Elements (page 622).

Mulit-Variate Results

- In the absence of Classification categories, the symbol used for a scatter plot is fixed and cannot be edited (a solid red aquare).

- If you defined more than one category, GoldSim will label the points in the Scatter Plots based on the categories you have defined. The Data Style for categories is defined at the bottom of the Monte Carlo Resilt Display Propertites dialog (in the section labled "Realization Calssification and Screening"). This dialog is accessed by pressing the **Options…** button in the Result Properties dialog of a Time History Result element.

  Read more: Using Result Classification and Screening in Multi-Variate Results (page 643).

Array Results

- The Data Style for the different array items is defined in the Array Label Set dalog.

  Read more: Controlling the Chart Style in Array Results (page 659).

Creating and Using Chart Styles

Often, after modifying the appearance of a chart, it is very useful to save the particular set of properties you have created (e.g., label fonts, header, footer) so that you can apply them to another chart. To facilitate this, GoldSim allows you

to save the set of properties as a named ***chart style***. A chart style consists of all of the chart properties that you specify in the Chart Style dialog.

Read more: Editing the Appearance of a Chart (page 661).

The named chart style is saved as part of the model file, and can subsequently be applied to other results in your model. You can also export and import chart styles between models.

Note: Unlike the other attributes of a chart (e.g., axes labels, headers, legends), the manner in which the data itself is shown in the chart is not a property of the chart itself (and hence is *not* edited in the Chart Style dialog and is *not* saved in a chart style.)

Read more: Editing Data Styles (page 669).

Whenever you view a result, it will initially take on a set of characteristics defined by the *default chart style* for that particular type of chart (e.g., time history, distribution, etc.).

GoldSim provides four "built-in" chart styles:

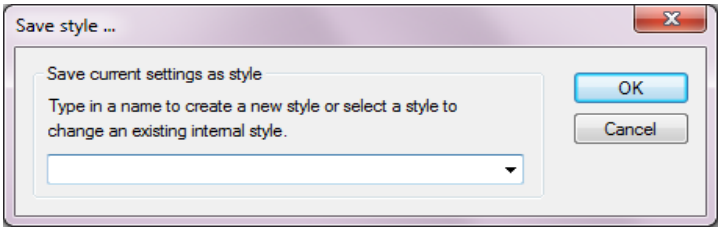
- Basic Time History Style
- Basic Distribution Result Style
- Basic Multi-Variate Style
- Basic Array View Style

Initially, the four "built-in" styles are defined as the default chart styles for each type of plot. For any type of chart, you can specify one of your own custom chart styles as the default chart style for all subsequent results that are displayed.

Saving and Applying Chart Styles

After you have customized the appearance of a particular result, you can save a new chart style as follows:

1. From the **General** tab of the Chart Style dialog, press the **Save Style…** button.
2. To save a style, press the **Save Style…** button. The following dialog is displayed:

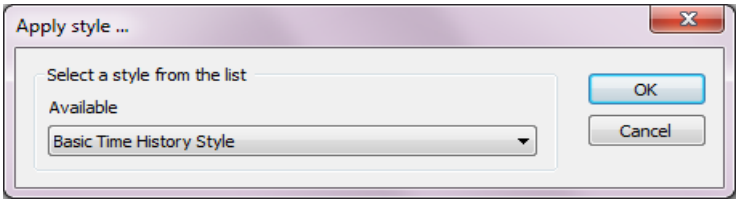


3. Type in a style name (to create a new style) or select an existing style from the drop list (to overwrite an existing style).
4. Press **OK** to save the style.

Note: You cannot overwrite the "built-in" styles provided by GoldSim. As a result, the drop list only contains user-defined styles.

You can apply a chart style to a result chart as follows:

1. From the **General** tab of the chart editing dialog, press the **Apply Style…** button.
2. The following dialog is displayed:



3. Select the style you wish to apply, and press **OK**.

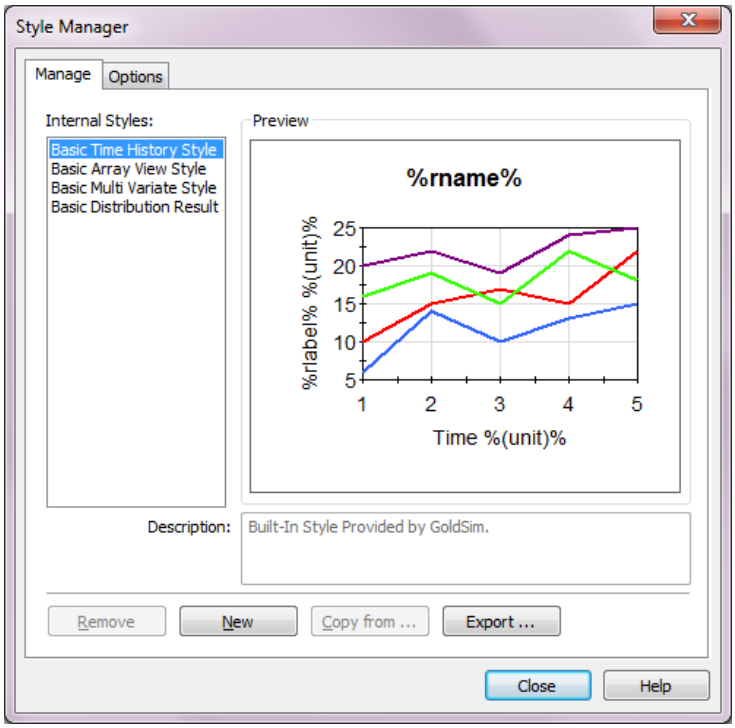
Note: Styles store all the information required for all types of charts (with the exception of the Data Style). Hence, any style can be applied to any kind of chart. In practice, however, you will typically create and apply different styles for different types of charts in order to better customize their appearance.

Using the Style Manager

***Style Manager** button*

The GoldSim Style Manager is used to manage all of your styles. You can use the Style Manager to view where styles are being used, create new styles, import and export styles, and to define default styles.

The Style Manager is accessed from the main menu via **View|Style Manager…** or by pressing the Style Manager button in the Standard toolbar.



The Style Manager consists of two tabs: **Manage** and **Options**. The **Manage** tab lists all of the styles in your model (including the "built-in" styles provided by GoldSim). When you select a style, a preview (using a simple default data set), showing its primary attributes is shown in the Preview window. You can also

enter a **Description** for any user-defined style (in order to remind yourself of the purpose or contents of the style).

You can delete a style by selecting it and pressing the **Remove** button. Note, however, that this button is grayed out if a "built-in" style is selected ("built-in" styles cannot be deleted).

If you right-click on one of the listed styles, a list of the Result elements which use that style is displayed.

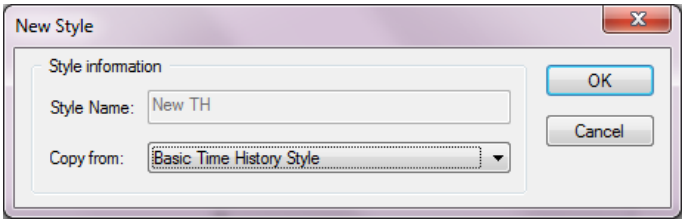
Read more: Creating and Using Result Elements (page 526).

Managing Styles in the Style Manager

You can use the Style Manager to copy the properties from one style to another, as well as import and export styles.

In some situations, you may have applied a style to a number of Result elements, and then want to edit that style so it is identical to some other style in your model. Rather than editing the style, you can simply copy all the properties of an existing style into another style as follows:

1. Press the **Copy from…** button from the **Edit Style** tab of the Style Manager. The following dialog is displayed:



2. Select an existing style from the **Copy from** drop-down list.
3. Press **OK**.

Chart styles are saved within the model file. Often, however, you may wish to use a style that you created in one model file within a second model file. To facilitate this, GoldSim allows you to save a style as an external ***style file***, and then create a new style (within a different model file) by importing the external style file.

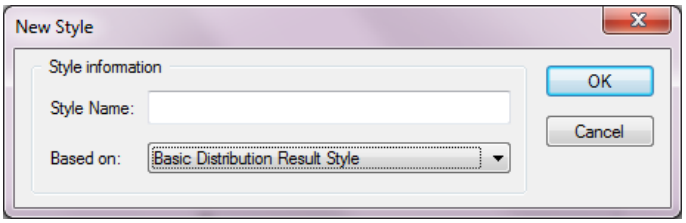
You can export a style (i.e., create a style file) as follows:

1. Press the **Export…** button from the **Manage** tab of the Style Manager. A Save As dialog will be displayed for saving the style to a file. The default filename is identical to that of the style. If you wish, you can modify it. Style files have the extension ".gcs".
2. Press the **Save** button to save the style file.

Note: The default location in which the style file will be saved is specified in the **Options** tab of the Style Manager. If desired, of course, you can also save the style file to a different folder.

You can import a style file into a model file as follows:

1. Make sure that the style file that you wish to import is in the default style file folder (specified in the **Options** tab of the Style Manager).
2. Press the **New…** button from the **Manage** tab of the Style Manager. The following dialog will be displayed:



3. If one of the built-in styles was selected when your pressed the **New** button, you can enter a **Style Name** for the style into which you wish to import the information contained in the style file (anda new style will be created). Othwerwise, it will be imported into the selected style (overwriting it).
4. Expand the **Based on** drop list to display all of the available styles upon which you can base the new style.
5. In addition to including all of the styles, the list also includes all the style files located in the default style file directory. (Style files include the date they were created.)
6. Select the style file that you wish to import and press **OK**.

The new style will be added to the style list. You may want to edit the **Description** for the new file to indicate how it should be used.

Defining the Default Chart Styles

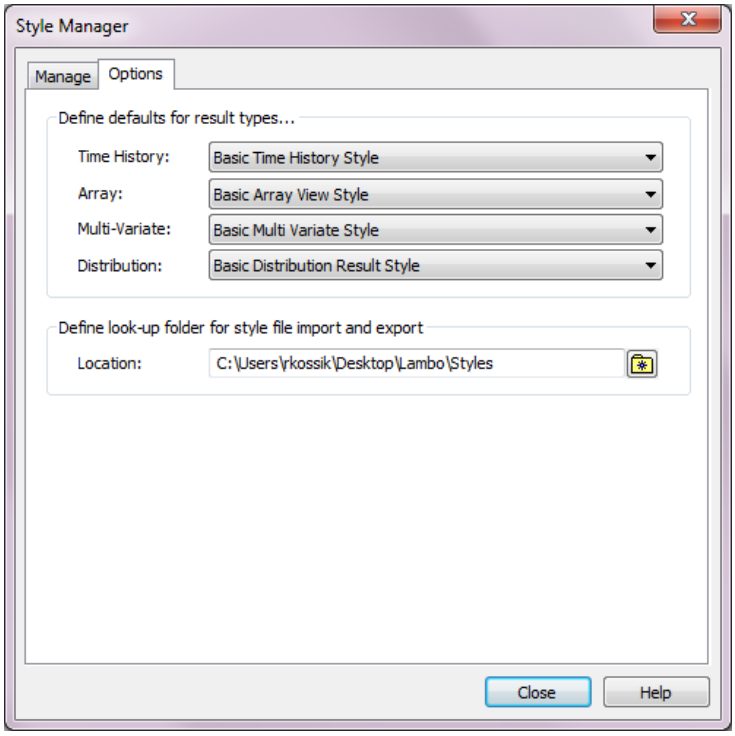
Whenever you view a result, it will initially take on a set of characteristics defined by the default chart style for that particular type of chart (e.g., time history, distribution, etc.).

GoldSim provides four "built-in" chart styles:

- Basic Time History Style
- Basic Distribution Result Style
- Basic Multi-Variate Style
- Basic Array View Style

Initially, these four "built-in" styles are defined as the default chart styles for each type of plot. For any type of chart, however, you can specify one of your own custom chart styles as the default chart style for all subsequent results that are displayed.

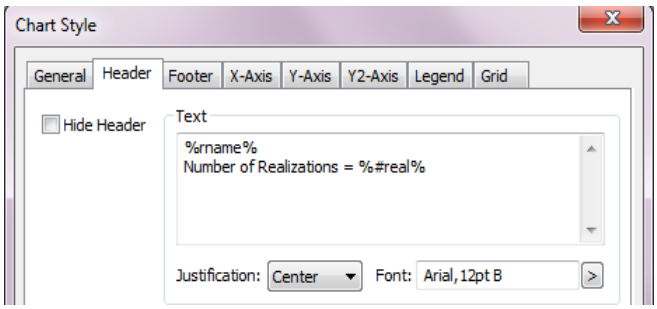
This is done from the **Options** tab of the Style Manager. The default styles for each of the four types of result charts are specified at the top of this tab of the dialog.



Note: Styles store all the information (except for Data Styles) required for all types of charts. Hence, any style can be applied to any kind of chart. In practice, however, you will typically create and apply different styles for different types of charts in order to better customize their appearance. Hence, it is likely that you will want to assign a different default chart style for each type of chart.

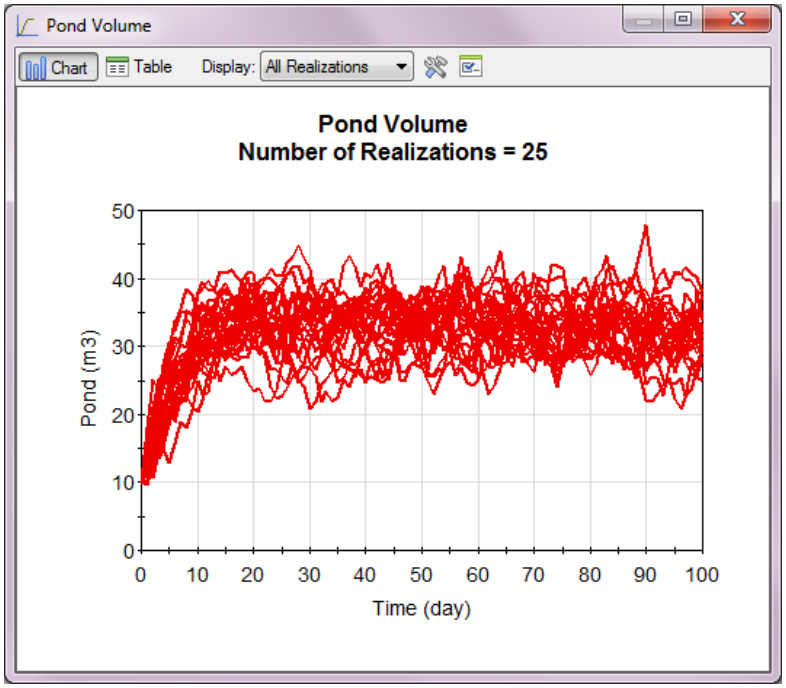
Using Keywords in Styles

In order to facilitate the use of a single chart style for multiple results, GoldSim allows you to use ***keywords*** (delimited by the % symbol) when you create and use chart styles. Keywords can be used in fields where titles or text is required (e.g., headers and footers, axis labels, legend titles). For example, the keyword %rname% inserts the name of the Result element being displayed, and the keyword %#real% inserts the number of realizations. We could insert these in a header, as shown below:



The keyword %rname% is actually the default header text for all results. It is hidden for interactive results (since there is no corresponding Result element).

This would appear in the chart as follows:



You can access the full list of keywords which are available by right-clicking in any Title or Text field in the chart style dialog.

A complete list of available keywords is provided below:

Keyword	Description
%rname%	Result element name[1]
%rdesc%	Result element description[1]
%rlabel%	Label of the first result mapped to the axis[2] (context sensitive)
%unit%	Display units for axis value[3] (context sensitive)
%(unit)%	Display units for axis value within parentheses[3] (context sensitive)
%[unit]%	Display units for axis value within brackets[3] (context sensitive)
%#real%	Number of realizations
%rundate%	Date simulation was started (displayed using Regional date format settings)
%runtime%	Time simulation was started (displayed using Regional time format settings)
%runduration%	Duration of simulation (simulation run time)
%curdate%	Current date (displayed using Regional date format settings)
%curtime%	Current time (displayed using Regional time format settings)
%period%	Period Label of reporting period being displayed (blank if result is not period-based)
%author%	Model Author (as specified in the Information tab of the Simulation Settings dialog)
%an_desc%	Analysis Description (as specified in the Information tab of the Simulation Settings dialog)
%filename%	model filename (without extension)
%filename_ex%	model filename (with extension)

Keyword	Description
%copyright%	GoldSim copyright notice
%version%	GoldSim version number

[1] *If an interactive result is being displayed, this is ignored.*

[2] *If the keyword is used in a location other than an axis title (e.g., a header or footer title), it simply appears as entered (e.g., %rlabel%) and is not replaced with a result label (as it would ambiguous as to which label to use).*

[3] *If the axis data is dimensionless, or if it represents a date/time, the keyword is ignored. If the keyword is used in a location other than an axis title (e.g., a header or footer title), it simply appears as entered (e.g., %unit%) and is not replaced with a unit (as it would ambiguous as to which unit to use).*

Note that some of these keywords are used as defaults. For example, all result displays use %rname% in the Header, and the axis titles use both %rlabel% and %(unit)%.

Exporting Results

In some cases, rather than using the result plotting and post-processing capabilities provided by GoldSim, you may wish to plot, analyze, or store the results using a separate program, such as a spreadsheet, a database, or statistical analysis package.

To facilitate this, GoldSim provides two flexible methods for exporting results in the form of an MS-Excel file or a text file. The two methods are as follows:

Exporting Time Histories from a Time History Result Element. You can export outputs that are specified within a Time History Result to an MS-Excel spreadsheet file or a text file. This export can be set to occur automatically at the end of a simulation, or can be triggered manually while in Result or Scenario Mode. Spreadsheet export provides export of a single history per result (for single realization runs) or a specified statistic (for multi-realization runs). Text export provides export of all realizations for multi-realization runs.

Exporting Results Using a Spreadsheet Element. You can export specified GoldSim outputs to an MS-Excel spreadsheet file using a Spreadsheet element. Any GoldSim output can be exported in this way. By using the built-in capabilities of Spreadsheet elements (in particular, the ability to specify dynamic offsets), time histories and/or multiple realizations can be output. This export occurs automatically during the simulation.

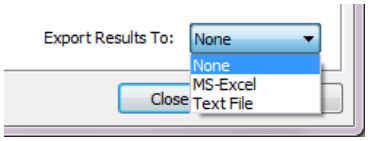
These methods are described in detail in the sections below.

Exporting from a Time History Result Element to a Spreadsheet

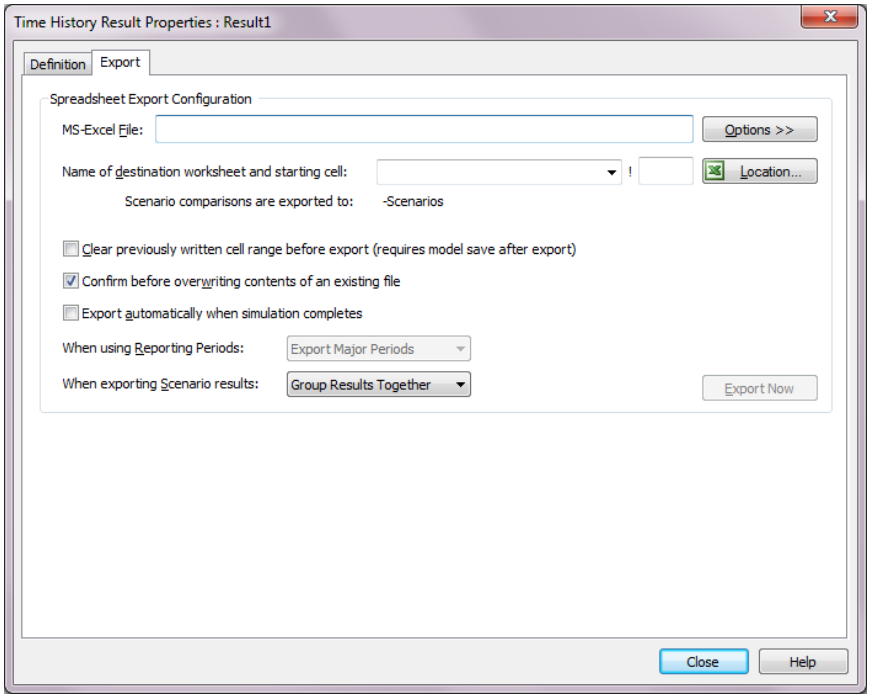
You can export outputs that are specified within a Time History Result to an MS-Excel spreadsheet file. This export can be set to occur automatically at the end of a simulation, or can be triggered manually while in Result or Scenario Mode. In addition, when running multiple scenarios, scenario comparisons can be written to the spreadsheet.

Read more: Exporting Scenario Results from a Time History Result Element to a Spreadsheet (page 684).

To enable this capability for a Time History Result element, select "MS-Excel" from the **Export Results To** drop-list at the bottom of the Time History Result element dialog:



By default, this is set to "None". When you select "MS-Excel", an **Export** tab is added to the dialog:



You must first specify the Excel filename to which you wish to export. You can enter the name of a Microsoft Excel spreadsheet file to which you wish to link by pressing the **Options >>** button. This will provide options for either selecting an existing file, or creating (and then selecting) a new file. You can also type in the name of a file directly. In this case, you must provide a full or relative path (relative to the model file).

The Excel file does not need to exist prior to the export. If you type in a file that does not exist, GoldSim will create it at the time of export. The filename that you enter can contain one or more of the following keywords which will be replaced with the appropriate text at the time of export:

Keyword	Description
%en%	The name of the Result element
%filename%	The name of the GoldSim file (without the extension)
%rundate%	Date simulation was started (displayed using Regional time format settings)
%runtime%	Time simulation was started (displayed using Regional time format settings).

If the file exists at the time of export, it will be overwritten by the new data. If **Confirm before overwriting contents of an existing file** is checked, GoldSim will confirm before doing so. Note, however, that for existing spreadsheet files,

GoldSim only overwrites those cells to which it is specifically exporting results. It does not overwrite any other cells in the file. In some cases, this could lead to situations where only a portion of the data in a particular row or column is overwritten (e.g., if the number of timesteps is changed between simulations), and this has the potential to create confusion. To prevent this, you should check **Clear previously written cell range before export**. If this is checked, GoldSim clears out all the cells from the previous time it exported from this Result element before writing the new results.

Note: In order to be able to clear cells that were previously written to, GoldSim must "remember' the cells to which it has previously written by saving this information with the file *after* the export has taken place. As a result, if you carry out an export, and then close the file *without first saving it*, this information is lost (i.e., GoldSim does not "remember" the cells to which it has previously written). Hence, to take advantage of this feature, you should always save the model after you carry out an export.

In addition to specifying the Excel filename, you must also specify the **Name of destination worksheet and starting cell** (the sheet and starting cell) where the data is to be exported. If the Excel file already exists, you can use the **Location…** button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell using your mouse. If the Excel file does not exist, you must manually enter the sheet name and starting cell.

Note: Scenario results are exported to a different sheet. In particular, the name of the sheet is "*Sheetname*-Scenarios", where *Sheetname* is the name that you specify in this dialog.

Read more: Exporting Scenario Results from a Time History Result Element to a Spreadsheet (page 684).

Data is always exported in columns (one column for the time, and one column for each result). If only a single realization is run, that realization is exported (for each result specified in the Result element). If multiple realizations are run, the specified **Custom Statistic** for each result is exported.

Read more: Viewing Custom Statistics for Multiple Realizations (page 560).

Note: If you wish to export multiple statistics, you can do so by simply adding multiple result items to the Result element that reference the same output but have different Custom Statistics.

Note: If you need to export all realizations of a multiple realization simulation, you can do so by choosing to export to a text file rather than a spreadsheet.

Read more: Exporting from a Time History Result Element to a Text File (page 686).

An example of what the exported data in a spreadsheet looks like is provided below:

	A	B	C
1	Date	Height	Flow_Rate
2		m	m3/day
3		Mean	50%
4	1/1/2013 0:00	10.01478	103.5608
5	1/2/2013 0:00	9.957983	102.2609
6	1/3/2013 0:00	9.876554	106.2593
7	1/4/2013 0:00	10.01512	96.57172
8	1/5/2013 0:00	10.03233	104.795
9	1/6/2013 0:00	9.852924	102.4378
10	1/7/2013 0:00	9.951	101.2296
11	1/8/2013 0:00	9.902326	101.7324
12	1/9/2013 0:00	10.10255	103.8019
13	1/10/2013 0:00	10.05865	99.36085
14	1/11/2013 0:00	10.005	98.30874

In this particular example, the Result element contained two results (Height and Flow_Rate). The model was run for multiple realizations, and hence the values that are written represent the Custom Statistic (which was Mean for Height and the 50th percentile for Flow_Rate).

Note that GoldSim writes three header rows. The first identifies the result, the second the units, and the third the result type (in this example, the Mean the the 50th percentile). In addition, the first cell contains a comment with information regarding the run (e.g., version used, date/time of simulation and export, name of Result element, etc.).

Note: GoldSim exports the results as single-precision values. When exporting dates from a calendar time model, if the first cell of the time column is not formatted as a date, GoldSim will format the column as a localized date (based on Windows settings). None of other columns (i.e., the result values) will be formatted.

The time points that are exported to the spreadsheet are the same as those displayed in result charts and tables for the result element.

Read more: Specifying When Time History Results Will Be Saved (page 423).

If your Result element is configured to display Reporting Periods, and you have more than one Reporting Period defined, you must specify which period you wish to export in the **When using Reporting Periods** drop-list. The type of Reporting Period results that are exported (e.g., Average, Cumulative, etc.) are as specified in the Properties dialog for the Result element.

Read more: Viewing Reporting-Period Based Results in Time History Result Elements (page 564).

If a result is an array, it is simply treated in the spreadsheet as separate items (i.e., an array of 10 items would produce 10 result columns).

Several other points regarding result export to spreadsheets should be noted:

- Conditions are exported as 0 (False) or 1 (True).
- Results that are dates (such as the output of a Milestone element) are exported as Julian days.
- If an output is an array, only those items of the array which have been selected to be viewed (via the Array Label Set dialog) are exported.

Read more: Viewing Time Histories for Array Outputs (page 547).

- For simulations involving multiple realizations, only those realizations which have been specifically selected to be included (via the Monte Carlo Result Options dialog) are used to compute the Custom Statistic that is exported.

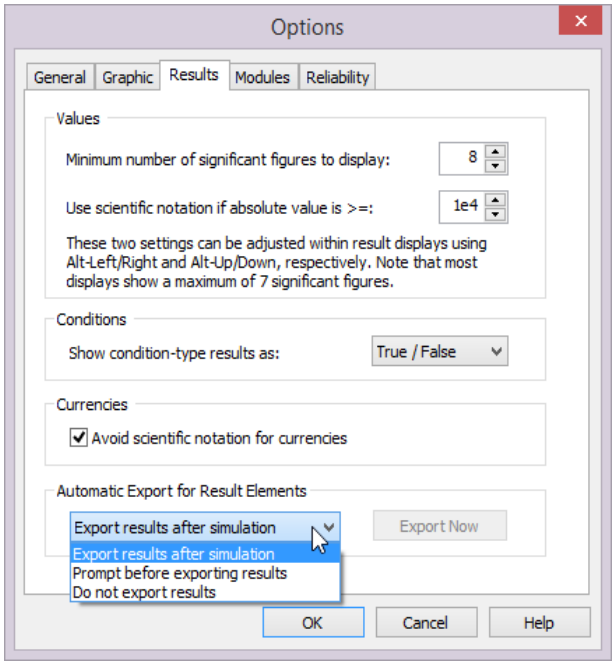
  Read more: Classifying and Screening Realizations (page 533).

- High resolution results are not exported. That is, if your Time History Result is configured to include unscheduled updates, these are not exported. Only scheduled updates are exported.

  Read more: Viewing Unscheduled Updates in Time History Result Elements (page 586).

The actual export of the results can be triggered in two ways: manually and automatically. This is determined by the **Export automatically when simulation completes** checkbox. If this box is cleared, GoldSim only exports the data if and when you press the **Export Now** button in the **Export** tab of the Result element dialog. If this box is checked, however, GoldSim automatically exports the results at the end of the simulation. When it exports the results in this way, any errors it encounters (e.g., running out of space in the spreadsheet) are written to the Run Log.

For elements which are set to export automatically, you can globally control their behavior from the **Results** tab of the Options dialog (accessed from the main menu via **Model|Options…**, and then selecting the **Results** tab):



The options in the drop-list affect all Time History Result elements which are set to export automatically when the simulation completes. If "Export results after simulation" is selected, any Result elements which are set to export results will export (silently) at the end of the simulation. If "Prompt before exporting results" is selected, at the end of the simulation, you will be prompted to determine whether or not to carry out an automatic export. If "Do not export results" is selected, no automatic export is carried out (i.e., this overrides the selection for each individual Result element).

Pressing the **Export Now** button manually exports results from all Time History Result elements.

Note: Within Dashboards, the Button control provides a command option that is equivalent to pressing the **Export Now** button from the **Result** tab of the Options dialog.

Button command options within a Dashboard are discussed in detail in the **Dashboard Authoring Module User's Guide**.

Several points should be noted regarding automatic export of spreadsheet results:

- Automatic export is ignored when carrying out a Sensitivity Analysis or an Optimization run.

  Read more: Running Sensitivity Analyses (page 497); Running an Optimization (page 486).

- Automatic export is ignored if the Result element is inside a SubModel.

  Read more: Using SubModels to Embed Models Within Models (page 914).

- Automatic export is only carried out under special circumstances when running and comparing scenarios.

  Read more: Exporting Scenario Results from a Time History Result Element to a Spreadsheet (page 684).

Finally, when exporting to spreadsheets, the following points regarding the use of Excel should be noted:

- GoldSim supports .xlsx, .xlsm, and .xls Excel files. However, if you have an older version of Excel (prior to Office 2007), you will need to install Microsoft's Office Compatibility Pack in order to read .xlsx and .xlsm files. Note that GoldSim does not officially support versions of Excel prior to Excel 2003.

- The .xls format is limited to 65,536 rows and 256 columns. Exporting in columns allows a maximum of 65,533 plot points to be exported (three rows are reserved for the header) and 255 output values (first column is reserved for time). If the data will not fit in the spreadsheet due to one of these limitations, a warning message will be written to the screen (if Exporting manually) or the Run Log (if exporting automatically). Note, however, that Excel 2007 and later support an extended worksheet size (1,048,576 rows by 16,384 columns). If you wish to export data to an extended worksheet range into GoldSim, you must use Excel 2007 or newer, and the file format must be .xlsx or .xlsm.

- You should never try to open a spreadsheet that is linked to GoldSim while a simulation is running. In addition, prior to running a GoldSim model that is linked to a spreadsheet, all referenced spreadsheets should be closed. If you try to interact with a spreadsheet that is being used by GoldSim during a simulation, all interaction with the spreadsheet is blocked by GoldSim. However, under some circumstances this may not be possible and could lead to errors. Hence, as a general rule, you

should not use Excel while a Goldsim model that references Excel is running.

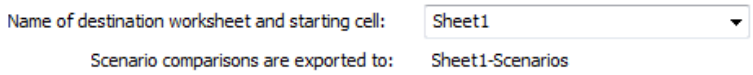
Exporting Scenario Results from a Time History Result Element to a Spreadsheet

When running multiple scenarios, time histories for different scenarios can be written to a spreadsheet.

Read more: Creating, Running and Comparing Scenarios (page 463).

Note, however, that export of scenario results to a spreadsheet must be carried out in a special way to prevent each export from simply overwriting previously exported scenarios.

In particular, when exporting scenarios, GoldSim exports all scenarios with results to a separate sheet from that defined in the dialog; it appends "-Scenarios" to the specified sheet name, and exports results to that sheet (creating it if it does not already exist). For example, if your specified sheet name was simply "Sheet1", scenario results would be exported to "Sheet1-Scenarios" (and this is indicated in the dialog):



Scenario results are exported only under the following four conditions:

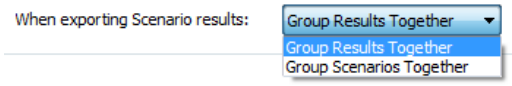
1. If the model is in Scenario Mode, pressing **Export Now** from the **Export** tab of the Result element exports results for all scenarios with results to the specified Scenarios sheet.
2. If the model is in Scenario Mode, pressing **Export Now** from the **Results** tab of the Model Options dialog exports results for all scenarios with results from *all* Time History Result elements to their specified Scenarios sheet.
3. Within Dashboards, the Button control provides a command option that is equivalent to pressing the **Export Now** button from the **Result** tab of the Options dialog. Hence, if the model is in Scenario Mode, this button command exports results for all scenarios with results from *all* Time History Result elements to their specified Scenarios sheet. Button command options within a Dashboard are discussed in detail in the **Dashboard Authoring Module User's Guide**.
4. If you press **Run All** in the Scenario Manager, any Result elements that are configured to automatically exports result will export results for *all scenarios* to their specified Scenarios sheet.

Several important points should be noted regarding scenario export:

- If you run a single scenario from the Scenario Manager or from the Scenario control on a Dashboard, no results are exported (even for Result elements that are configured to automatically export).
- If you run a model such that it ends in Result Mode (e.g., by pressing **F5** or the **Run** button on the Run Controller), any Result elements that are configured to automatically export results will simply export results for the Active Scenario to the selected primary sheet (e.g., Sheet1). Similarly, if you press **Export Now** in Result Mode, the results for the Active Scenario will be exported to the primary sheet. In either case, any other results in the primary sheet will be overwritten. As a result, *you cannot export scenario comparisons in this way.*
- Since Player users cannot access the Scenario Manager, and single scenario runs are never automatically exported, if a Scenario control is

present on a Dashboard when running the Player, automatic export will always be ignored. Results can only be exported manually. To export scenarios from the Player, the author should provide a Button control with an Export command (item #3 above). Note that the Result tab of the options dialog is accessible from the Player, so if the model is browsable, scenario export could also be carried out by pressing the Export Now button in that dialog (item #2). However, this would typically be a very awkward way for a Player user to trigger an export. The Scenario control within a Dashboard is discussed in detail in the **Dashboard Authoring Module User's Guide**.

When exporting scenario results to the Scenarios sheet GoldSim provides two options for how the results are exported on the **Export** tab of the Result element dialog:



If you select "Group Results Together", if multiple results are being exported, the columns will be grouped together by result:

	A	B	C	D	E	F	G
1		Scenario 1	Scenario 2	Scenario 3	Scenario 1	Scenario 2	Scenario 3
2	Time	Flow	Flow	Flow	Cost	Cost	Cost
3	day	m3/day	m3/day	m3/day	$/day	$/day	$/day
4		#1	#1	#1	#1	#1	#1
5	0	14.09524155	26.0219841	40.11722565	1390.157349	3664.960205	3791.338379
6	1	14.53176975	26.82788277	41.35965347	2129.185059	5613.306641	5806.868164
7	2	13.41220188	24.76099014	38.17319107	2075.336914	5471.343262	5660.010254
8	3	12.42288685	22.93456268	35.35744858	2014.817627	5311.791992	5494.95752
9	4	11.57713223	21.37316704	32.95029831	1134.683472	2991.438232	3094.591553
10	5	12.16874218	22.46537018	34.63411331	2095.246338	5523.831055	5714.308105
11	6	11.62099266	21.45414162	33.07513428	1578.418457	4161.285156	4304.777832
12	7	16.18246269	29.87531662	46.05778122	1773.099243	4674.53418	4835.724609
13	8	11.67915154	21.56151009	33.24066162	1365.847046	3600.869385	3725.037598
14	9	13.03471088	24.06408119	37.09879303	1893.643677	4992.333008	5164.482422
15	10	11.18868828	20.65603828	31.84472466	1796.660522	4736.650391	4899.983398

In this particular example, the Result element contained two results (Flow and Cost) and three scenarios (Scenario1, Scenario2, and Scenario3). Note that all the results for Flow are presented (for each scenario), followed by all the results for Cost.

If you select "Group Scenarios Together", if multiple results are being exported, the columns will be grouped together by scenario:

	A	B	C	D	E	F	G
1		Scenario 1	Scenario 1	Scenario 2	Scenario 2	Scenario 3	Scenario 3
2	Time	Flow	Cost	Flow	Cost	Flow	Cost
3	day	m3/day	$/day	m3/day	$/day	m3/day	$/day
4		#1	#1	#1	#1	#1	#1
5	0	14.09524155	1390.157349	26.0219841	3664.960205	40.11722565	3791.338379
6	1	14.53176975	2129.185059	26.82788277	5613.306641	41.35965347	5806.868164
7	2	13.41220188	2075.336914	24.76099014	5471.343262	38.17319107	5660.010254
8	3	12.42288685	2014.817627	22.93456268	5311.791992	35.35744858	5494.95752
9	4	11.57713223	1134.683472	21.37316704	2991.438232	32.95029831	3094.591553
10	5	12.16874218	2095.246338	22.46537018	5523.831055	34.63411331	5714.308105
11	6	11.62099266	1578.418457	21.45414162	4161.285156	33.07513428	4304.777832
12	7	16.18246269	1773.099243	29.87531662	4674.53418	46.05778122	4835.724609
13	8	11.67915154	1365.847046	21.56151009	3600.869385	33.24066162	3725.037598
14	9	13.03471088	1893.643677	24.06408119	4992.333008	37.09879303	5164.482422
15	10	11.18868828	1796.660522	20.65603828	4736.650391	31.84472466	4899.983398

Note that in this case all the results for the first scenario are presented (for each result), followed by all the results for the second scenario, and finally all the results for the third scenario.

Note that when exporting scenario results, GoldSim writes four header rows (rather than three). The first identifies the scenario, the second the result, the third the units, and the fourth the result type (in this example, Realization #1).

In addition, the first cell contains a comment with information regarding the run (e.g., version used, date/time of simulation and export, name of Result element, names of scenarios being exported, etc.).

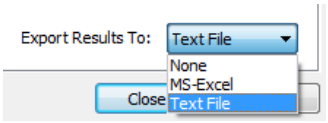
Exporting from a Time History Result Element to a Text File

You can export outputs that are specified within a Time History Result to a tab-delimited text file. This export can be set to occur automatically at the end of a simulation, or can be triggered manually while in Result Mode. Unlike spreadsheet exports, when running multiple realizations, instead of outputting only a single statistic for each result, in additinn, all realizations can be exported to the text file.

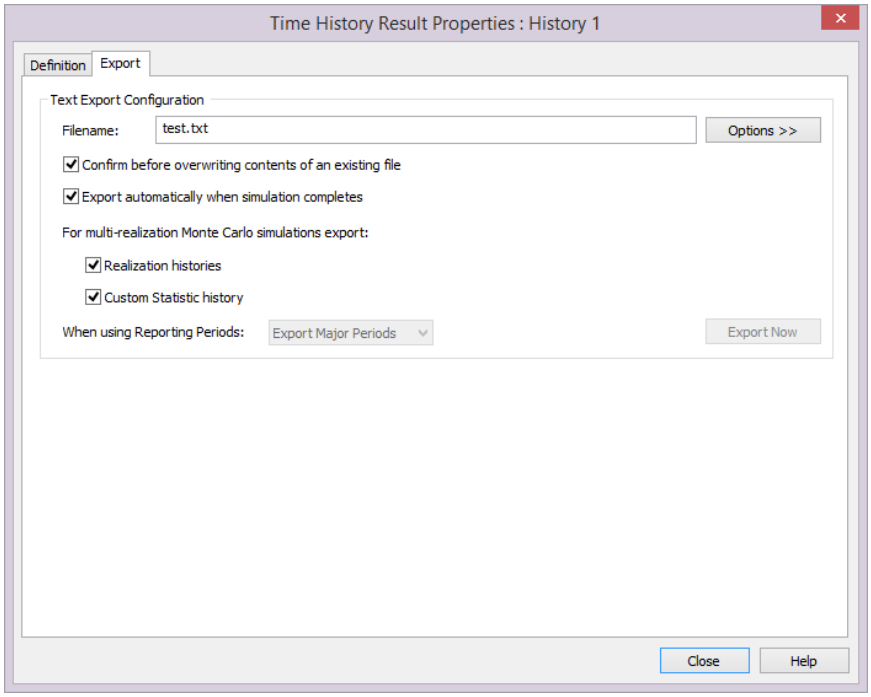
Note: You cannot export scenario results to a text file. If you need to export scenario results, you can do so by choosing to export to a spreadsheet rather than a text file.

Read more: Exporting Scenario Results from a Time History Result Element to a Spreadsheet (page 684).

To enable this capability for a Time History Result element, select "Text File" from the **Export Results To** drop-list at the bottom of the Time History Result element dialog:



By default, this is set to "None". When you select "Text File", an **Export** tab is added to the dialog:



You must first specify the text filename to which you wish to export. You can enter the name of an existing file to which you wish to link by pressing the **Options >>** button. This will provide options for either selecting an existing text file (with the extension .txt), or creating (and then selecting) a new file. You can also type in the name of a file directly. In this case, you must provide a full or relative path (relative to the model file).

Note: The output file must have the extension ".txt".

The file does not need to exist prior to the export. If you type in a file that does not exist, GoldSim will create it at the time of export. The filename that you enter can contain one or more of the following keywords which will be replaced with the appropriate text at the time of export:

Keyword	Description
%en%	The name of the Result element
%filename%	The name of the GoldSim file (without the extension)
%rundate%	Date simulation was started (displayed using Regional time format settings)
%runtime%	Time simulation was started (displayed using Regional time format settings).

If the file exists at the time of export, it will be overwritten by the new data. If **Confirm before overwriting contents of an existing file** is checked, GoldSim will confirm before doing so.

You then must specify what you would like to export. If running a single realization, the options are nor relevant. However, when running multiple realizations, you can export two kinds of information:

For multi-realization Monte Carlo simulations export:

☑ Realization histories

☑ Custom Statistic history

You can export the individual histories for each result and/or you can export the Custom Statistic for each result.

Note: If you wish to export multiple statistics, you can do so by simply adding multiple result items to the Result element that reference the same output but have different Custom Statistics.

Data is always exported in a tab-delimited format, with header information written at the top of the file. If multiple realizations are run, all (unscreened) realizations are written to the file. Multiple results are exported in separate blocks (unless the results reference the same output, have the same label and the same axis mapping, in which case they are exported in the same block so as to list all results referencing the same output together).

An example of what the exported data in a text file looks like is provided below:

```
!GoldSim Version: 11.0.0000
!Model File: C:\Users\rkossik\Desktop\test.gsm
!Simulation Time: 7/7/2013 11:11
!Export Time: 7/7/2013 11:11
!Result Name: Result1
!Result Path: \
!Result Description: (none)
!Time Points: 5
!Realizations: 5
!Weights:       1       1       1       1       1

!Result: Flow
!Unit: m3/day
(Date)  #1      #2      #3      #4      #5
"1/1/2013 0:00:00"      10.70646        11.28938        8.448304        9.233838        9.802564
"2/1/2013 0:00:00"      11.17828        10.51371        10.19991        9.988766        10.40429
"3/1/2013 0:00:00"      10.31708        9.398015        10.06617        11.69179        10.22887
"4/1/2013 0:00:00"      9.556067        8.286284        9.691882        11.48532        11.37942
"5/1/2013 0:00:00"      8.905486        11.67070        8.445362        10.52193        9.000431

!Result: Cost
!Unit: $/day
(Date)  #1      #2      #3      #4      #5
"1/1/2013 0:00:00"      1852.756        1198.883        1217.685        1481.143        1687.330
"2/1/2013 0:00:00"      1935.623        1548.398        1061.841        1280.337        1269.063
"3/1/2013 0:00:00"      1886.670        1384.743        1554.021        1775.878        1997.357
"4/1/2013 0:00:00"      1831.652        1768.296        1494.343        1293.594        1959.220
"5/1/2013 0:00:00"      1031.530        1268.548        1339.408        1547.809        1219.714
```

In this particular example, the Result element contained two results (Flow and Cost). The model was run for 5 realizations, and contains 5 timesteps. If you export both realizations and one or more statistics, for each result, the realizations are listed first, followed by columns for the statistics.

Note that the first 10 lines contain header information (each preceded by a "!"):

- Version number
- Filename
- Time simulation was run
- Time export took place
- Name of Result element
- Full path to Result element
- Description for Result element
- The number of time points
- The number of realizations

- The relative realization weights (only included if realization are exported)

The time points that are exported to the text file are the same as those displayed in result charts and tables for the result element.

Read more: Specifying When Time History Results Will Be Saved (page 423).

The relative realization weights are normally all 1. However, if you use importance sampling (e.g., for Stochastics or Timed Events) or directly specify weights in the Simulation Settings dialog, they will be different and it is critical to take this into account for any post-processing that you carry out on results.

Read more: Applying Importance Sampling to a Stochastic Element (page 182); Timed Event Elements (page 333); Probabilistic Simulation Options (page 439).

After the header information, the result data is written. For each result, there are two header rows (preceded by a "!") indentifying the result name and the units. The realization numbers are then shown, followed by a row for each timepoint. Each row contains all the (unscreened) realizations.

Note: GoldSim exports the results as single-precision values.

If your Result element is configured to display Reporting Periods, and you have more than one Reporting Period defined, you must specify which period you wish to export in the **When using Reporting Periods** drop-list. The type of Reporting Period results that are exported (e.g., Average, Cumulative, etc.) are as specified in the Properties dialog for the Result element.

Read more: Viewing Reporting-Period Based Results in Time History Result Elements (page 564).

If a result is an array, it is simply treated in the text file as separate items (i.e., an array of 10 items would produce 10 result groupings).

Several other points regarding result export to text files should be noted:

- Conditions are exported as 0 (False) or 1 (True).
- All dates are exported wrapped in quotes.
- If an output is an array, only those items of the array which have been selected to be viewed (via the Array Label Set dialog) are exported.

  Read more: Viewing Time Histories for Array Outputs (page 547).

- For simulations involving multiple realizations, only those realizations which have been specifically selected to be included (via the Monte Carlo Result Options dialog) are exported.

  Read more: Classifying and Screening Realizations (page 533).

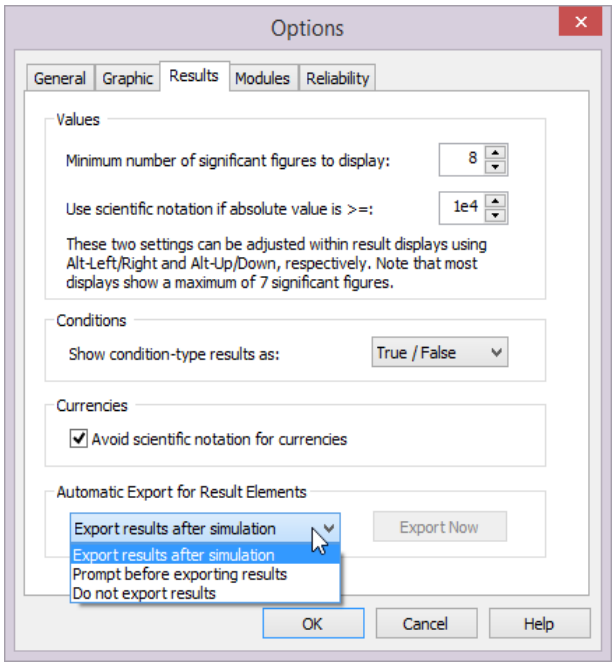
- High resolution results are not exported. That is, if your Time History Result is configured to include unscheduled updates, these are not exported. Only scheduled updates are exported.

  Read more: Viewing Unscheduled Updates in Time History Result Elements (page 586).

The actual export of the results can be triggered in two ways: manually and automatically. This is determined by the **Export automatically when**

simulation completes checkbox. If this box is cleared, GoldSim only exports the data if and when you press the **Export Now** button in the **Export** tab of the Result element dialog. If this box is checked, however, GoldSim automatically exports the results at the end of the simulation.

For elements which are set to export automatically, you can globally control their behavior from the **Results** tab of the Options dialog (accessed from the main menu via **Model|Options…**, and then selecting the **Results** tab):



The options in the drop-list affect all Time History Result elements which are set to export automatically when the simulation completes. If "Export results after simulation" is selected, any Result elements which are set to export results will export (silently) at the end of the simulation. If "Prompt before exporting results" is selected, at the end of the simulation, you will be prompted to determine whether or not to carry out an automatic export. If "Do not export results" is selected, no automatic export is carried out (i.e., this overrides the selection for each individual Result element).

Pressing the **Export Now** button manually exports results from all Time History Result elements.

Note: Within Dashboards, the Button control provides a command option that is equivalent to pressing the **Export Now** button from the **Result** tab of the Options dialog.

Button command options within a Dashboard are discussed in detail in the **Dashboard Authoring Module User's Guide**.

Several points should be noted regarding automatic export of text file results:

- Automatic export is ignored when carrying out a Sensitivity Analysis or an Optimization run.

  Read more: Running Sensitivity Analyses (page 497); Running an Optimization (page 486).

- Automatic export is ignored if the Result element is inside a SubModel.

  Read more: Using SubModels to Embed Models Within Models (page 914).

- Automatic export is ignored when running and comparing scenarios.

Exporting Results Using a Spreadsheet Element

An additional method that can be used to export results from GoldSim is to use a Spreadsheet element. A spreadsheet element can be used to export any output to a spreadsheet. By using the built-in capabilities of Spreadsheet elements (in particular, the ability to specify dynamic offsets), time histories, scenarios, and/or multiple realizations can all be output.

Copying and Exporting Result Displays

Once you have created some result displays (in the form of charts or tables), you may want to copy and/or export the results. This is described below.

Copying a Chart or Table

GoldSim allows you to copy a chart or table to the clipboard. You can then subsequently paste it into another application (e.g., a word processor). Note that charts are pasted as enhanced metafiles, so that the picture can be edited after it is pasted into another application (if you have software that allows you to edit an enhanced metafile).

To copy a chart display window, right-click in the chart, and select **Copy** from the context menu. Alternatively, you can press **Ctrl+C** while viewing the chart.

Read more: Using Context Menus in Charts (page 522).

Table data can be copied using **Ctrl+C**. Note, however, that before copying a table, you must select the items of the table that you wish to copy. If your cursor is simply placed in a cell of the table, only that cell will be copied to the clipboard. You can select multiple cells in the table by placing your cursor in one cell and dragging to another location in the table. Selected items will be indicated in black. Double-clicking on the item in the upper left-hand corner of the table selects the entire table.

Read more: Selecting Items and Copying Values in Result Tables (page 523).

Once you have copied multiple items from a table, you can readily paste them into a spreadsheet. The table retains its rows and columns in the clipboard, and therefore will be correctly pasted into separate rows and columns in the spreadsheet.

Note: If you press the Copy button (or **Ctrl+C**) from the Summary View of a Distribution result, the preview pane chart is copied to the clipboard.

Exporting a Chart

You can export a chart to a file of a specified format. You can then subsequently open and edit the file in a separate application (e.g., a graphics package).

To export a chart, right-click in the chart, and select **Export…** from the context menu.

Read more: Using Context Menus in Charts (page 522).

You will be prompted for the graphics format. The following choices are available:

- Windows Bitmap (BMP)

- Enhanced Windows Metafile (EMF)
- JPEG Image (JPEG)
- Portable Network Graphics (PNG)
- Graphics Interchange Format (GIF)

Chapter 9: Documenting and Presenting Your Model

A mathematical theory is not to be considered complete until you have made it so clear that you can explain it to the first man whom you meet on the street.

David Hilbert

Chapter Overview

A model which cannot be easily explained is a model that will not be used or believed. As a result, GoldSim was specifically designed to allow you to effectively document, explain and present your model. You can add graphics, explanatory text, notes and hyperlinks to your model, and organize it in a hierarchical manner such that it can be presented at an appropriate level of detail to multiple target audiences. This chapter describes how to use these important features.

In this Chapter

The following topics are discussed in this chapter:

- Step One: Organize Your Model!
- Displaying Your Model in a Presentation Using Full Screen View
- Adding Graphics and Text
- Modifying the Appearance of Elements
- Creating, Editing and Viewing Notes
- Adding Hyperlinks to the Graphics Pane
- Manipulating Graphical Objects
- Creating Printed Documentation
- Creating a GoldSim Player File

Step One: Organize Your Model!

The most important way to ensure that your model is easy to explain and present is to spend some time to organize your model.

The sections below provide some guidance and ideas for ensuring that your model is well organized.

Defining the Audiences for Your Model

When you begin to design your model, you should always have in mind the audiences to whom you will be presenting or explaining the model. Most models will have at least two audiences:

- a high-level audience (managers and decision-makers who are primarily interested in the "big picture" or "bottom line"); and
- a low-level audience (analysts who are also interested in the technical details and assumptions of the model).

Often, there will be multiple low-level audiences, which are interested in different technical aspects of the model. There may also be a "mid-level" audience, which is interested in more than the "big picture", but does not want to get lost in the details.

For a model to be successful (i.e., useful), you must be able to present and explain your model effectively to all of these audiences.

GoldSim provides the tools for you to create a single model that can be explained and presented to multiple audiences. The primary way in which you accomplish this is by organizing your model into a logical, top-down hierarchy.

Creating a Top-Down Model

Perhaps the most important requirement of a well-documented and easily understood model is that it be well-organized. The primary characteristic of a well-organized model is that it is structured in a “top-down” manner.

Containers provide the mechanism in GoldSim by which you can create well-organized, top-down models. You do this by placing model elements in a top-down containment hierarchy in which the level of detail increases as you "push down" into the hierarchy.

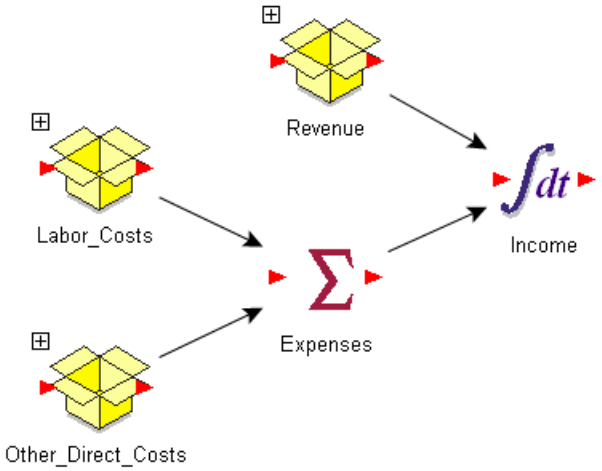
In a top-down model, you can imagine that each Container has a specific "message" and a corresponding audience at which it is targeted.

For example, a model's highest level Container might have the following message: "This is the problem I am trying to solve, and here is the overall structure of the model I have built to solve it". Such a Container would be intended for all audiences.

Another Container's message in the model might be "Here are the detailed assumptions regarding process X". This Container's audience would be the analysts or technical personnel interested in that particular aspect of the model.

You "hide" these details in a Container, because they are of no interest to higher level audiences (e.g., managers). Those who are interested, however, can be shown the contents of the Container.

As a simple example of this, consider the model below (a financial model for a company):



This high-level container shows the "big picture" of how the company's future income is predicted by the model, and would be of interest to all audiences. The three Containers (Revenue, Labor_Costs, and Other_Direct_Costs) contain the details of these three model components. The details within the Revenue Container may be of particular interest to one individual (e.g., the sales manager), while the contents of the other Containers may be of particular interest to others (e.g., cost accountants). Others (e.g., financial analysts, the CFO) may be interested in the details of all three.

Note: Even if your model was not originally organized in a top-down manner, you can use GoldSim's Move function to add Containers and move elements into them after your model has been built.

Read more: Understanding Containers (page 100); Moving Elements Between Containers (page 106).

Other Suggestions for Organizing Your Model

In addition to structuring your model in a top-down manner, the following suggestions can also help you to create well-organized and easy to understand models:

- Name the elements of your model carefully. If possible, avoid using abbreviations that will not be easily recognized by others.
- Use the Description field for all of your elements. Because the Description is displayed in tool-tips, it can be an effective and highly visible documentation tool.
- Don't hide the details of your model within long input expressions. It is better to add a few additional elements such that the relationships in the model can be shown graphically and are transparent.
- Filter influences in your model if they detract from the presentation. It is often particularly useful to filter influences at the highest model level (to make the diagrams easier to view).

  Read more: Filtering Influences (page 392).
- Modify the color of influences to highlight key relationships.

  Read more: Editing the Appearance of Influences (page 389).

- In some cases, it may be worthwhile to place similar items in a single Container (even if they are referenced throughout the model). For example, you may want to place all constant inputs in one Container named "Constants" or "Key_Assumptions". If it is important that some of these elements also appear in the Container in which they are referenced (in order to add clarity), you could place a *clone* of the element in that Container.

  Read more: Cloning Elements (page 893).

- It is often useful to create a "Results" Container in the model Root (or perhaps one containment level down) with Result elements for all of the model's key outputs. This allows you to quickly access and display the model's results.

  Read more: Creating and Using Result Elements (page 526).

- You can use the Hyperlink object to provide navigation aids for your model. The Hyperlink object can be used to provide links "jumps" to specified Containers, Dashboards or elements from any location.

  Read more: Adding Hyperlinks to the Graphics Pane (page 712).

Displaying Your Model in a Presentation Using Full Screen View

GoldSim provides a Full Screen View to facilitate the presentation of your model.

***Full Screen View** button*

Full Screen View expands the graphics pane to fill the full screen (and hides all toolbars, menus and browsers). Full Screen View is activated by clicking the **Full Screen View** button, by selecting **View|Full Screen View** from the main menu, or by pressing **F11**.

When in Full Screen View, all toolbars, menus and browsers are hidden *and cannot be activated*. You can, however, turn the *Note pane* on and off using **Ctrl-Shift-N** or by clicking on a Note hyperlink in the model.

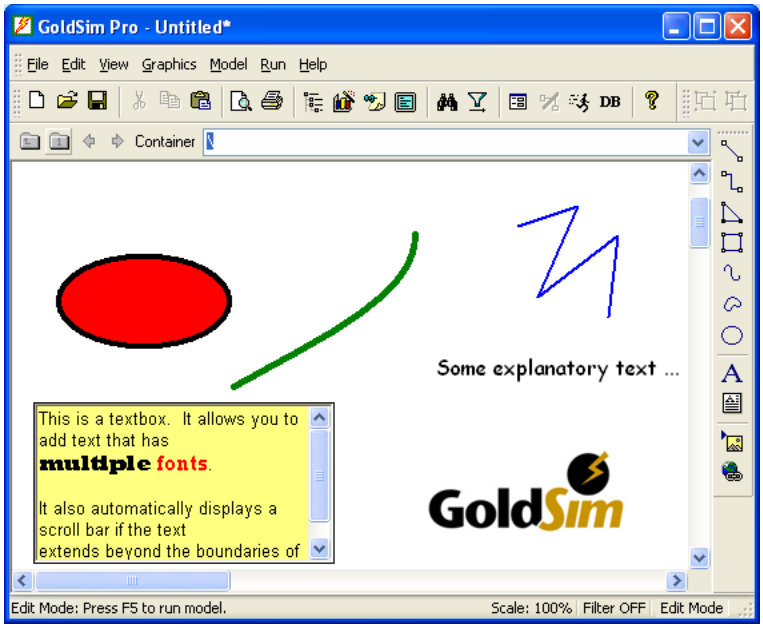
Read more: Creating, Editing and Viewing Notes (page 709).

Within Full Screen View, the **Leave Full Screen** button is displayed in a small box in the upper right hand corner of the screen. The box can be moved or closed (by clicking on the x).

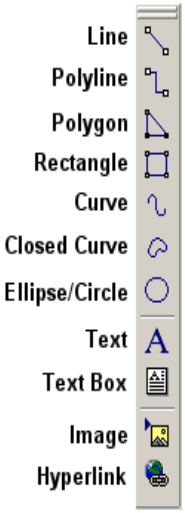
You can exit Full Screen View by clicking on the **Leave Full Screen** button, pressing **Esc**, or pressing **F11**.

Adding Graphics and Text

Adding graphics (an image or a simple schematic) and explanatory text to the graphics pane is one of the most powerful and straightforward ways to document and explain a model.



The Drawing Tools toolbar in GoldSim provides access to a number of buttons for adding graphics and text to your model. The Drawing Tools toolbar is present by default to the right of the graphics pane (although it can be moved to another location):



The same functions can also be accessed under **Graphics|Insert** in the main menu.

Adding and editing graphic objects, text and images is discussed in the sections below.

Adding Graphic Objects

Graphic objects can be added to the graphics pane by using the Drawing Tools toolbar (originally docked to the right of the graphics pane), or by selecting **Graphics|Insert** from the main menu.

\+

Drawing cursor

All of the graphic object buttons (or menu options) operate in the same way: when you select a button or a menu option, the cursor changes its appearance (to a drawing cursor). Clicking in the graphics pane with the cursor allows you to insert the object at that point.

You can delete a graphic object by selecting it and pressing the **Delete** button, or by right-clicking on it and selecting **Delete** from the context menu.

Adding a Line

To add a Line to the graphics pane:

1. Select **Line** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want the line to start.
3. Left-click and drag the line to the location where you want it to end.

You can move a line by selecting it and dragging it around the graphics pane. You can move one end of the line by selecting the line and dragging one of the end points (vertices).

You can rotate a line by selecting the line and choosing **Graphics|Rotate** from the main menu.

Read more: Rotating Objects (page 720); Changing the Appearance of Graphic Objects (page 700).

Adding a Polyline

A Polyline is a segmented line. To add a Polyline to the graphics pane:

1. Select **Polyline** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want the polyline to start, and click.
3. Move the cursor to the next vertex of the polyline and click.
4. After adding as many vertices as desired, move the cursor to the location where you want the polyline to end and double-click.

You can move a polyline by selecting it and dragging it around the graphics pane. You can move a vertex of the polyline by selecting the line and dragging one of the vertices.

You can rotate a polyline by selecting the object and choosing **Graphics|Rotate** from the main menu

Read more: Rotating Objects (page 720); Changing the Appearance of Graphic Objects (page 700).

Adding a Polygon

To add a Polygon to the graphics pane:

1. Select **Polygon** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want one of the corners of the polygon to be and click.
3. Move the cursor to the next vertex of the polygon and click.
4. After adding as many vertices as desired, move the cursor to the location where you want the last vertex of the polygon to be and double-click.

You can move a polygon by selecting it and dragging it around the graphics pane. You can "stretch" a polygon by selecting the object and dragging one of the "handles" (there are eight handles - one on each corner and one on each side). Pressing the **Shft** key while resizing the object keeps the same aspect ratio; pressing the **Ctrl** key while resizing keeps the center of the object at the same location.

You can rotate a polygon by selecting the object and choosing **Graphics|Rotate** from the main menu.

Read more: Rotating Objects (page 720); Changing the Appearance of Graphic Objects (page 700).

Adding a Rectangle

To add a Rectangle to the graphics pane:

1. Select **Rectangle** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want one of the corners of the rectangle to be.
3. Left-click and drag the line to the location of the opposite corner of the rectangle.

You can move a rectangle by selecting it and dragging it around the graphics pane.

You can "stretch" a rectangle by selecting the object and dragging one of the "handles" (there are eight handles - one on each corner and one on each side). Pressing the **Shft** key while resizing the object keeps the same aspect ratio; pressing the **Ctrl** key while resizing keeps the center of the object at the same location.

You can rotate a rectangle by selecting the object and choosing **Graphics|Rotate** from the main menu.

Read more: Rotating Objects (page 720); Changing the Appearance of Graphic Objects (page 700).

Adding a Curve or a Closed Curve

To add a Curve or a Closed Curve to the graphics pane:

1. Select **Curve** or **Closed Curve** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want the curve to start, and click.
3. Move the cursor to the next "handle" of the curve and click.
4. After adding as many handles as desired, move the cursor to the location where you want the curve to end and double-click.

If you are drawing a closed curve, GoldSim will automatically draw a line from the end point to the starting point of the curve (in order to close the curve).

You can move a curve by selecting it and dragging it around the graphics pane. You can stretch the curve by selecting one of the "handles" and dragging it.

You can rotate a curve by selecting the object and choosing **Graphics|Rotate** from the main menu.

Read more: Rotating Objects (page 720); Changing the Appearance of Graphic Objects (page 700).

Adding an Ellipse/Circle

To add an Ellipse/Circle to the graphics pane:

1. Select **Ellipse/Circle** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want one of the corners of a rectangle surrounding the ellipse/circle to be.
3. Left-click and drag the line to the location of the opposite corner of the rectangle.

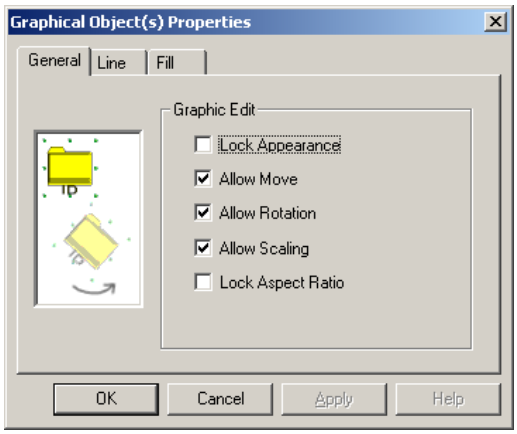
You can move an ellipse/circle by selecting it and dragging it around the graphics pane. You can "stretch" an ellipse/circle by selecting the object and dragging one of the "handles" (there are eight handles - one on each corner and one on each side). Pressing the **Shft** key while resizing the object keeps the same aspect ratio; pressing the **Ctrl** key while resizing keeps the center of the object at the same location.

You can rotate an ellipse/circle by selecting the object and choosing **Graphics|Rotate** from the main menu.

Read more: Rotating Objects (page 720); Changing the Appearance of Graphic Objects (page 700).

Changing the Appearance of Graphic Objects

To edit the properties of a graphic object, right-click on it (to access its context menu), and select **Appearance…**. A dialog like this will be displayed:



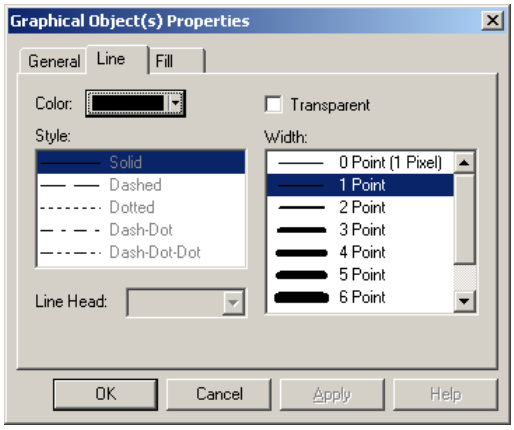
*The **Fill** tab is not present when editing lines, polylines, and curves.*

If the **Allow Move** and **Allow Rotation** boxes are checked (the default), you can move and rotate the object.

If **Allow Scaling** is checked (the default), you can resize the object (this option is not available for lines, polylines and curves). For objects that can be scaled, if **Lock Aspect Ratio** is checked, the object keeps its aspect ratio when it is resized.

If **Lock Appearance** is checked, all of these options are disabled (you cannot move, rotate or scale the object).

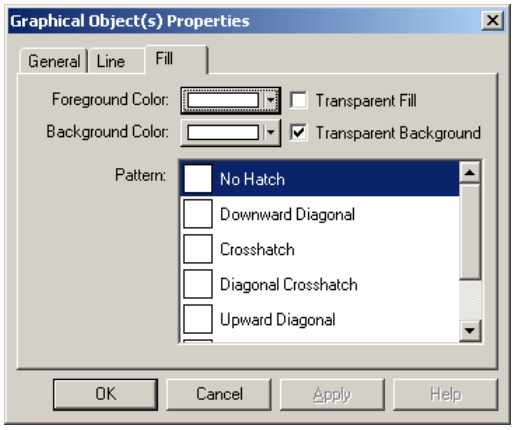
The **Line** tab allows you to edit the appearance of the line itself (for lines, polylines and curves) or the outline for the object (for polygons, rectangles, closed curves, and ellipses). You can select the **Color**, the **Style** and the **Width** (and for Lines and Polylines, the **Line Head**).



You can change the line's color, style and width. However, the style can only be applied to 0 point lines.

Note: The default Line Head for new Lines and Polylines can be defined in the Graphic tab of the Options dialog (accessed via **Model|Options** from the main menu).

The **Fill** tab (which is not available for lines, polylines and curves) allows you to change the appearance of the fill for the object.



The fill consists of a background and a foreground. The **Foreground Color** is superimposed on top of a **Background Color** according to a selected **Pattern**. By default, the background is transparent.

Adding and Editing Text

Text can be added to the graphics pane by using the Drawing Tools toolbar (originally docked to the right of the graphics pane), or by selecting **Graphics|Insert|Text** from the main menu.

text cursor

When you select the **Text** button or menu option, the cursor changes to the text cursor. Clicking in the graphics pane with the cursor allows you to insert the text object at that point.

Note: GoldSim provides two ways to add text to the graphics pane: a Text object and a Text Box. Text objects are movable objects that can utilize a single font. Text Boxes are scrollable boxes of text that can simultaneously utilize multiple fonts and can be "pinned" to a specific location in the graphics pane.

Read more: Adding and Editing Text Boxes (page 704).

To add a Text object to the graphics pane:

1. Select **Text** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want the text to be inserted and click.

"<Add your text here.>" is inserted into the graphics pane.

The default font which is used can be selected from the **Graphic** tab of the Options dialog (accessed via **Model|Options** in the main menu).

You can move a text object by selecting it and dragging it around the graphics pane. You can rotate a text object by selecting it and choosing **Graphics|Rotate** from the main menu.

You can delete a text object in the graphics pane by selecting it and pressing the **Delete** button, or by right-clicking on it and selecting **Delete** from the context menu.

Read more: Rotating Objects (page 720).

Editing Text Objects

Double-clicking on a text object in the graphics pane selects the text (it will be highlighted in blue). If you start typing while the entire text is highlighted in blue, the text will be overwritten. Once the text is selected, clicking anywhere within the text places the cursor at that location.

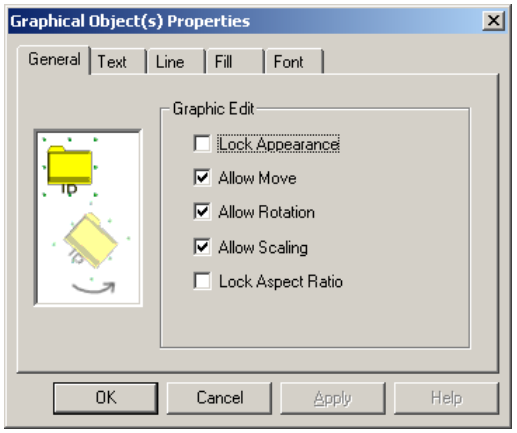
You can also use the **Home**, **End**, **Backspace** and arrow keys to move the cursor in the selected text.

A single click on a text object selects the object, but does not select the text itself. Eight "handles" will be visible, defining the limits of the *textbox* (one at each corner, and one on each side).

By default, word-wrap is on, so the text will wrap around given a specified width. As you type, the textbox will automatically expand downward.

Changing the Appearance of Text Objects

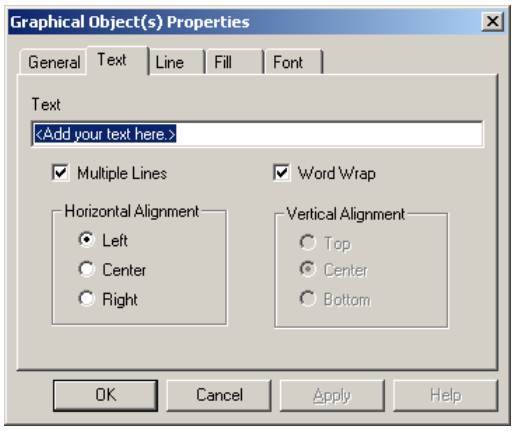
To edit the properties of a text object (e.g., font, fill), right-click on it (to access its context menu), and select **Appearance…**. The following dialog will be displayed:



If **Allow Move** and **Allow Rotation** boxes are checked (the default), you can move and rotate the text. If **Allow Scaling** is checked (the default), you can resize the textbox.

If **Lock Aspect Ratio** is checked, the textbox keeps its aspect ratio when it is resized. If **Lock Appearance** is checked, all of these options are disabled (you cannot move, rotate or scale the text).

The **Text** tab allows you to edit the text and adjust its alignment**.**

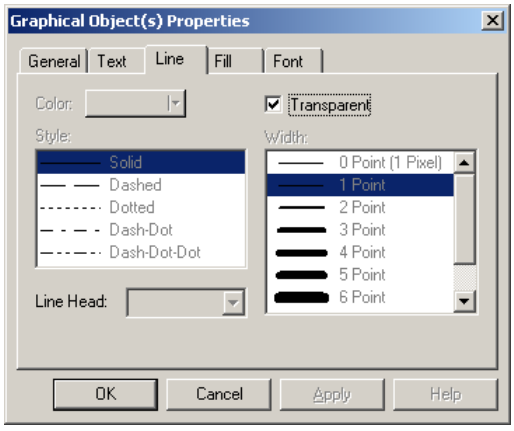


You can edit the **Text** directly in this tab. If **Multiple Lines** is checked (the default), the text object can continue onto multiple lines (either by pressing Enter while editing it or by turning on Word-wrap). The **Word Wrap** checkbox activates word wrap.

Warning: Word Wrap is ignored if **Multiple Lines** is turned off (cleared). That is, your text will only wrap if you allow for multiple lines.

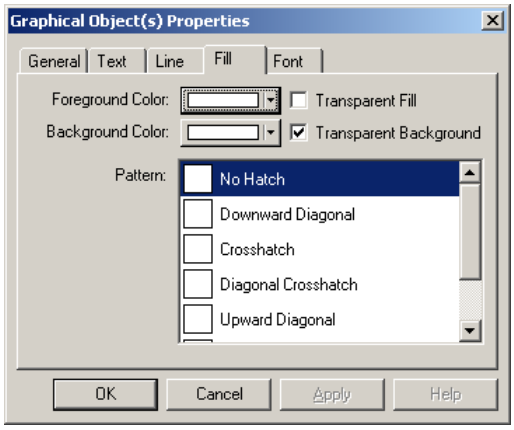
You can center the text horizontally in the textbox by choosing a **Horizontal Alignment** option. You can center the text vertically in the textbox by choosing a **Vertical Alignment** option. However, only single line text can be vertically aligned (the option is not available if **Multiple Lines** is checked).

The **Line** tab allows you to edit the appearance of the outline for the textbox.



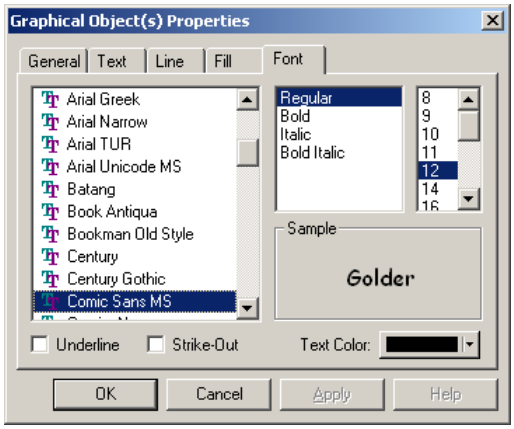
You can change the outline's **Color**, **Style** and **Width**. Note, however, that the style can only be applied to 0 point lines.

The **Fill** tab allows you to change the appearance of the fill for the textbox.



The fill consists of a background and a foreground. The **Foreground Color** is superimposed on top of a **Background Color** according to a selected **Pattern**.

The **Font** tab controls the font of the text.



Adding and Editing Text Boxes

Text boxes can be added to the graphics pane by using the Drawing Tools toolbar (originally docked to the right of the graphics pane), or by selecting **Graphics|Insert|Text Box** from the main menu.

Text box cursor

When you select the **Text Box** button or menu option, the cursor changes to the text box cursor. Clicking in the graphics pane with the cursor allows you to insert the text object at that point.

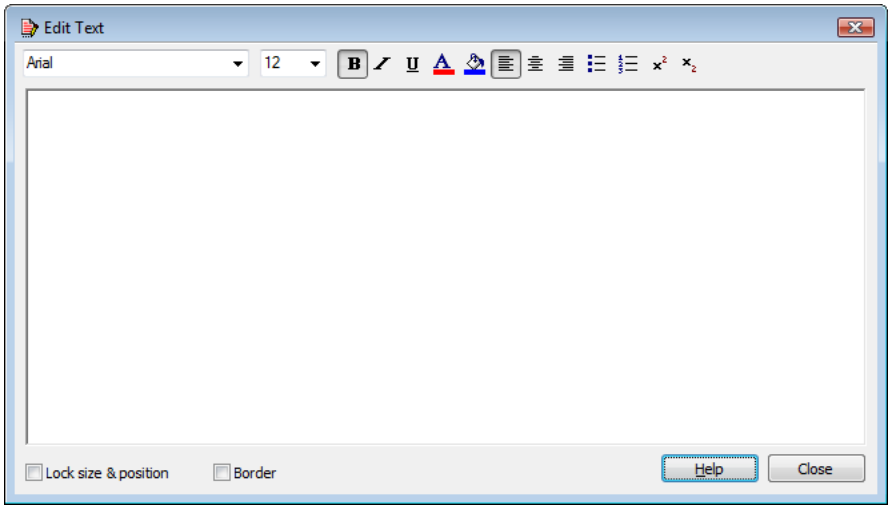
Note: GoldSim provides two ways to add text to the graphics pane: a text object and a text box. Text objects are movable objects that can utilize a single font. Text boxes are scrollable boxes of text that can simultaneously utilize multiple fonts and can be "pinned" to a specific location in the graphics pane.

Read more: Adding and Editing Text (page 701).

To add a text box to the graphics pane:

1. Select **Text Box** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want the text box to be inserted, left-click, drag the cursor to specify the size of the box, and release the mouse button.

At this point the text box will be inserted into the graphics pane and the dialog for editing the text box will be displayed:



After adding text, pressing the **Close** button closes the dialog.

You can double-click on the dialog to edit it again. You can move a text box by selecting it and dragging it around the graphics pane. You can resize the text box by grabbing and dragging one of the handles of the box. You can delete a text box in the graphics pane by selecting it and pressing the **Delete** button, or by right-clicking on it and selecting **Delete** from the context menu.

Note: All of these operations (editing, moving, resizing, deleting) can only occur if the size and position are not locked (i.e., the **Lock size & position** checkbox is cleared).

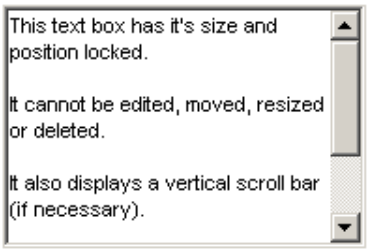
Controlling the Behavior of Text Boxes

You can control the format of the text in a text box by using the buttons and controls at the top of the dialog. These buttons and controls are self-explanatory (and all have tool-tips), and allow you to control the font, size, color, and justification of the text. Note that the text can have multiple fonts, formats and colors. You can also control these options by right-clicking on the text and selecting options from a context menu.

Note: The default font used within the text box can be selected from the **Graphic** tab of the Options dialog (accessed via **Model|Options** in the main menu).

If the **Border** checkbox is checked, a border is placed around the textbox.

The **Lock size & position** checkbox switches the text box between display mode and edit mode. If this box is checked, a vertical slider appears in the text box (if necessary), and you cannot move, resize, edit or delete the text box:



If the **Lock size & position** checkbox is cleared, the text box can be edited by double-clicking on it, can be moved by selecting and dragging, can be resized by dragging the handles, and can be deleted by selecting and pressing the **Delete** key. A text box which is not locked appears in the graphics pane like this:

You can unlock a text box in the graphics pane by right-clicking on it and clearing the **Lock size & position** menu item. Likewise, you can lock a text box in the graphics pane by right-clicking on it and checking the **Lock size & position** menu item.

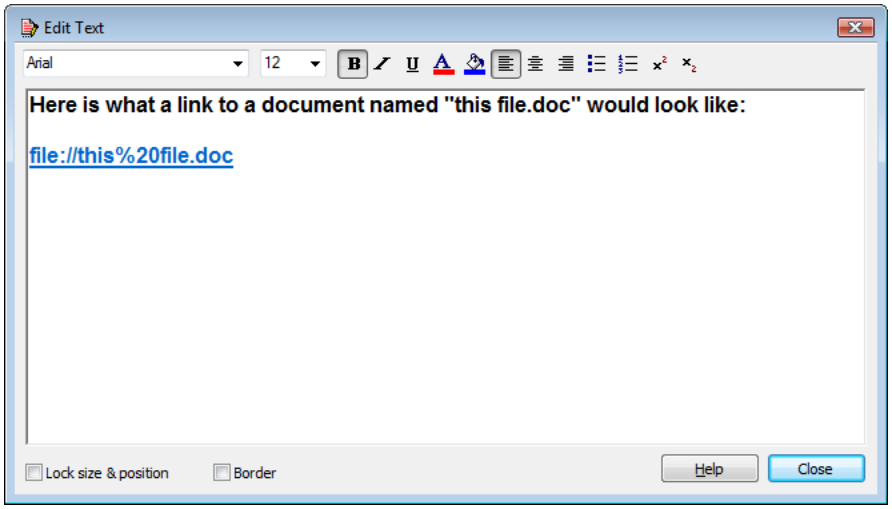
Inserting Hyperlinks into Text Boxes

You can insert hyperlinks from a text box to other documents:

- To insert a link to a URL which starts with "www", you need only type in the address (e.g., www.goldsim.com).
- To insert a link to a URL which does not start with "www", you must specify the transfer protocol (e.g., http://website.com/).
- To insert a link to an ftp site, you need only type in the address (e.g., ftp.asite.com).
- To insert a link that launches an email, you must type “mailto:” before the email address (e.g., mailto:name@company,com).
- To insert a link to some other document (e.g., a Word document or a PowerPoint presentation), you must type "file:///" before the document name and one of the following:
  - The full path (e.g., file://C:\Projects\Docs\report.doc).
  - A relative path (e.g., file://..\Docs\report.doc), in which case GoldSim searches relative to the current directory (the directory from which the model file was loaded). If the model file was in C:\Projects\Models, GoldSim would look for the document in C:\Projects\Docs.
  - No path (e.g., file://report.doc), in which case GoldSim looks only in the current directory (the directory from which the model file was loaded).

When you are entering a hyperlink, GoldSim recognizes it as a link as soon as it sees a space (it will become underlined and the text will become blue). As a

result, when you are entering the text, *there cannot be any spaces in the link*. If the filename includes a space, you can represent this by using the characters %20 to represent the space. For example, if the filename you wanted to link to was "this file.doc", the link in the Note pane would need to look like this:



When you click on a hyperlink in a text box, GoldSim will immediately open the application associated with the link (e.g., Internet Explorer, Word).

Adding Images

In addition to adding text and graphics, you can also add images (as bitmaps and other graphic formats) to the graphics pane.

To add an image to the graphics pane:

1. Select Image from the Drawing Toolbar or the main menu under **Graphics|Insert**. A dialog is displayed for selecting the image file to be inserted.
2. Select a file and file type to insert and press **OK**.
3. Place the cursor at the location in the graphics pane where you want the center of the image to be inserted and click.

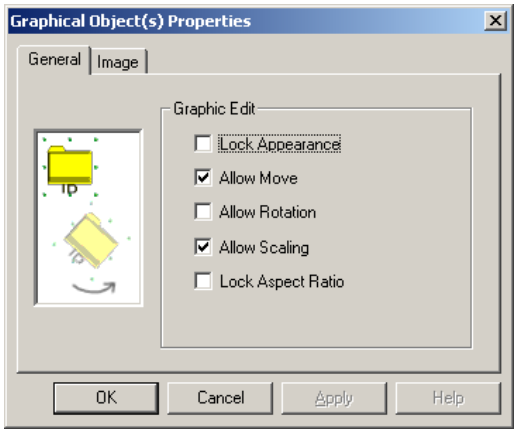
The image will be inserted into the graphics pane. You can move an image by selecting it and dragging it around the graphics pane.

You can "stretch" (resize) an image by selecting it and dragging one of the "handles" (there are eight handles - one on each corner and one on each side). Pressing the **Shft** key while resizing the image keeps the same aspect ratio; pressing the **Ctrl** key while resizing keeps the center of the object at the same location.

Changing the Appearance of an Image

GoldSim provides some capabilities for you to edit the properties and appearance of an image that you have inserted into the graphics pane.

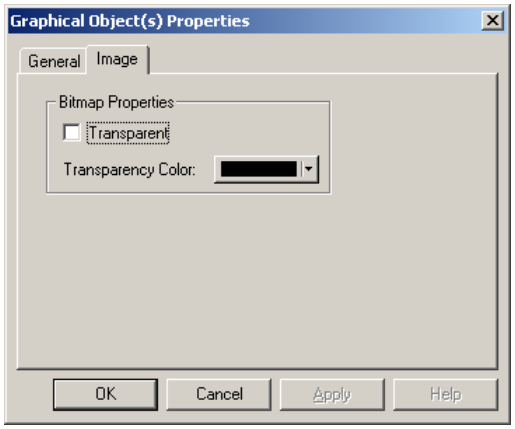
To edit the properties of an image in the graphics pane, right-click on it (to access its context menu), and select **Appearance…**. The following dialog will be displayed:



If the **Allow Move** box is checked (the default), you can move the image.

If **Allow Scaling** is checked (the default), you can resize the image. If **Lock Aspect Ratio** is checked, the image keeps its aspect ratio when it is resized (scaled). If **Lock Appearance** is checked, both of these options are disabled (you cannot move or scale the image).

The **Image** tab allows you to make one color in the image transparent. If you check the **Transparent** box, and choose a **Transparency Color**, that color will become transparent in the image.



Note: The Image tab is only available for bitmap and icon files. If your image is an enhanced metafile, the image tab is replaced with a Line and a Fill tab. The Line tab allows you control the appearance of the outline of the image. The Fill tab allows you to control the appearance of the background of the image (including allowing you to make it transparent).

Modifying the Appearance of Elements

All of the GoldSim elements have default symbols (graphical images by which the element is represented in the graphics pane).

Read more: GoldSim Element Types (page 77).

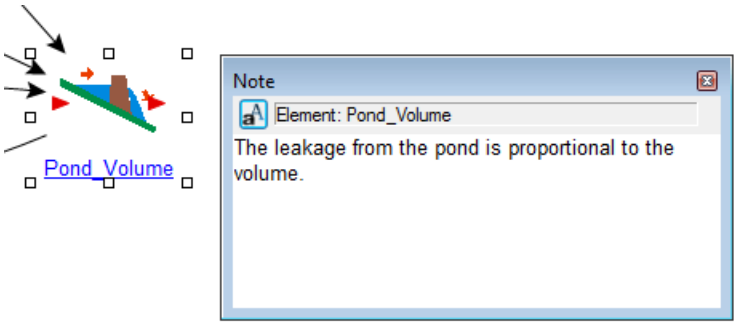
One of the most powerful ways to document and visually augment your model is to replace the default symbols with customized symbols. This is particularly

useful for Containers. For example, you may want to replace the image of a particular Container with a schematic or diagram of the subsystem that it represents.

Read more: Editing the Appearance of Elements (page 395).

Creating, Editing and Viewing Notes

In order to internally document your model, GoldSim allows you to attach a Note to each element:



The topics below describe how to create, edit and display notes.

Opening the Note Pane


***Note Pane** button*

To display, create and edit notes, you must activate the ***Note pane***. By default, the Note Pane is turned off (hidden). The Note pane can be toggled on and off in order to edit and view notes for your elements.

You do this by pressing the Note Pane button in the Standard toolbar, by selecting **View|Note** from the main menu, or selecting **Note** from the toolbar context menu (accessed by right-clicking anywhere outside of the graphics pane and browsers).

The Note pane is originally docked at the bottom of the screen. You can undock the pane and dock it at another location on the screen (e.g., the bottom, or to the right of the graphics pane) by grabbing the **docking grab bar** on the left side of the pane (a vertical bar) and *dragging* the pane to the desired location. If you press the **Ctrl** key while you drag the Note pane, it will become a floating window. You can also undock the pane (and turn it into a floating window) by double-clicking on the docking grab bar.

Whenever you select an element, the Note for that particular element is displayed in the Note pane. The element whose Note is being displayed is indicated in the Note pane. If nothing is selected in the graphics pane, the Note for the entire Container is displayed.

You can close the Note pane by pressing the Note Pane button in the Standard toolbar, by de-selecting **View|Note** from the main menu, by hitting the **Esc** key, or by pressing the Close button in the upper right-hand corner of the Note window.

Creating and Displaying Notes

To add a note to an element, do the following:

1. Select the element (by clicking on it).
2. Open the Note Pane by pressing the Note Pane button in the Standard toolbar or by selecting **View|Note** from the main menu.
3. Click on the Note Pane and start typing.

When you run GoldSim for the first time and you activate the Note pane, it will be docked at the bottom of the screen.

By default, the Format Toolbar is turned off (and a default font is selected). The default font for a new note is specified in the **Graphic** tab of the Options dialog (accessed via **Model|Options** in the main menu).

The Format Toolbar provides tools for formatting your Note. You can show and/or hide the Format Toolbar by pressing the Show Format Bar button in the upper lefthand corner of the Note pane.

Note: The basic settings for the Note pane (its location, size and whether or not the Format Toolbar is visible) is saved to the Windows Registry, and are therefore applied to all GoldSim files on your machine. Formatting for specific Notes (e.g., font, color), however, is saved only with the file.

You can only enter a Note if an element is selected (or nothing is selected). If nothing is selected, the Note is attached to the Container being viewed. If a graphical object is selected, you will not be able to type in the Note pane (since Notes can only be associated with elements).

In addition to typing in the Note pane, you can paste text from another Note or another application (such as a word processor).

Note: Notes in GoldSim support multibyte characters sets (for supporting character sets, such as Japanese and Chinese, that cannot be represented in a single byte).

Once a Note has been created for an element, the element's ID will be underlined in the Graphics Pane and a single click on the element ID in the Graphics Pane will open the Note pane and display the Note for that element:

By default, the ID of an element with a Note associated with it is underlined *and* shown in blue (so that it looks like a hyperlink). You can modify this so that the ID is shown in the defined text color (black by default) via an option on the **General** tab of the Options dialog (accessed by selecting **Model|Options** from the main GoldSim menu).

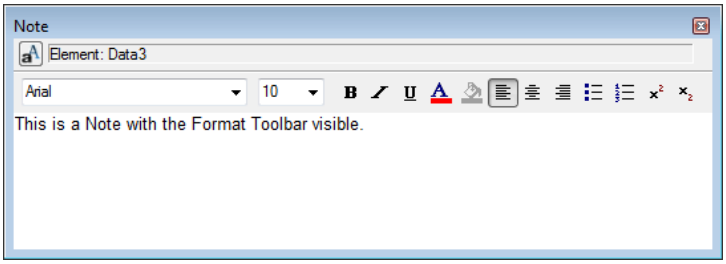
To delete a note attached to an element, simply delete all of the text in the note.

Editing and Formatting Notes

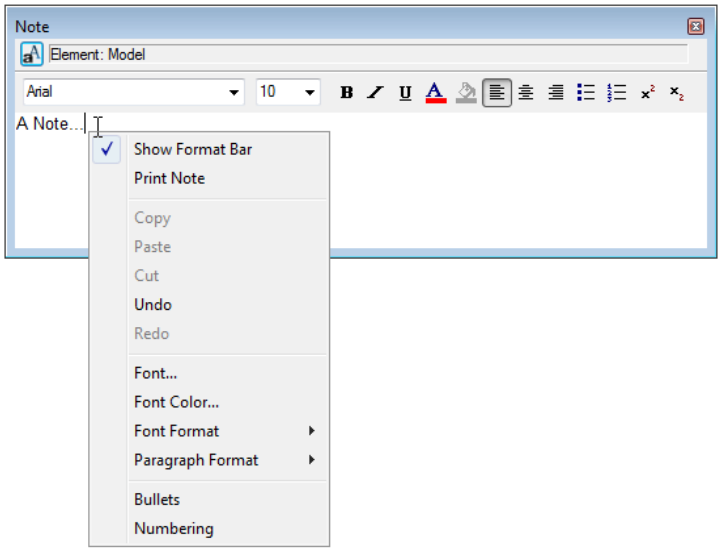
The Note Pane acts like a simple text editor. That is, you can specify alignment, fonts, font format, text color, indents, and create simple numbered and bulleted lists.

You can access formatting tools for your note in two ways:

- You can show and/or hide the Format Toolbar by pressing the Show Format Bar button in the upper lefthand corner of the Note pane:



- You can view a context menu with all formatting options by right-clicking anywhere within the Note pane:



The various formatting options in the toolbar and menu are self-explanatory. Note you can use this menu to Print the note.

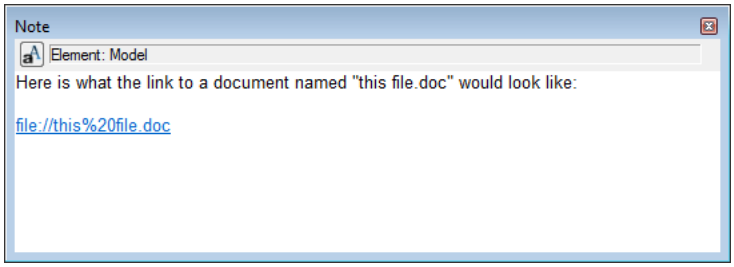
Inserting Hyperlinks into Notes

One of the most powerful features of Notes is that you can insert hyperlinks to other documents:

- To insert a link to a URL which starts with "www", you need only type in the address (e.g., www.goldsim.com).
- To insert a link to a URL which does not start with "www", you must specify the transfer protocol (e.g., http://website.com/).
- To insert a link to an ftp site, you need only type in the address (e.g., ftp.asite.com).
- To insert a link that launches an email, you must type “mailto:” before the email address (e.g., mailto:name@company,com).
- To insert a link to some other document (e.g., a Word document or a PowerPoint presentation), you must type "file:///" before the document name and one of the following:
  - The full path (e.g., file://C:\Projects\Docs\report.doc).
  - A relative path (e.g., file://..\Docs\report.doc), in which case GoldSim searches relative to the current directory (the directory from which the model file was loaded). If the model file was in C:\Projects\Models, GoldSim would look for the document in C:\Projects\Docs.
  - No path (e.g., file://report.doc), in which case GoldSim looks only in the current directory (the directory from which the model file was loaded).

When you are entering a hyperlink, GoldSim recognizes it as a link as soon as it sees a space (it will become underlined and the text will become blue). As a result, when you are entering the text, *there cannot be any spaces in the link*. If the filename includes a space, you can represent this by using the characters

%20 to represent the space. For example, if the filename you wanted to link to was “this file.doc”, the link in the Note pane would need to look like this:



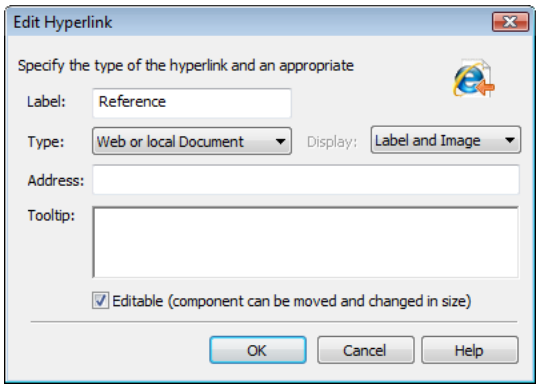
When you click on a hyperlink in a note, GoldSim will immediately open the application associated with the link (e.g., Internet Explorer, Word).

Adding Hyperlinks to the Graphics Pane

You can also add Hyperlink objects to the graphics pane. When you double-click on a hyperlink in the graphics pane, it can open another application (e.g., a website, a PowerPoint presentation, an email program, a Word document, a graphics image), or jump to a specific location in your model (which can be useful for navigation of complex models).

To insert a hyperlink, press the **Hyperlink** button on the Drawing Tools toolbar, or select **Graphics|Insert|Hyperlink…** from the main menu.

The following dialog will be displayed:



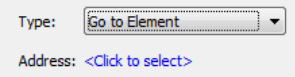
The **Label** appears below the Hyperlink object in the graphics pane. Note that unlike element IDs, the label can have spaces (but it cannot be completely empty).

The **Type** field contains the type of hyperlink you wish to make in the **Address** field. GoldSim provides the following options:

- **Web or local document**: Opens a website or a document (e.g., a Word or PowerPoint document);
- **Email to…**: Opens your default email program and inserts the specified email address in the “To” field. ;
- **FTP**: Opens an FTP site;
- **Execute Command**: Opens a specific program;

- **Go to Element**: Jumps to a specific element and selects it. This is useful for navigating a model.
- **Open Container**: Jumps to a specific Container. This is useful for navigating a model.
- **Open Dashboard**: Jumps to a specific Dashboard. This is useful for navigating a model.

In the latter three cases, after selecting the option, you must specify the element, Container or Dashboad that you wish to link to:



You can also enter a **Tooltip**. This is displayed when you place your cursor over the Hyperlink object in the graphics pane.

If the **Editable** check box is checked (the default), you will be able to move the Hyperlink object around the graphics pane (and resize it). If this checkbox is cleared, the object will be "pinned" to the location where you placed it. You can make the object editable by right-clicking on it in the graphics pane and checking the **Editable** menu item. Likewise, you can lock (make uneditable) an editable hyperlink by right-clicking on it in the graphics pane and clearing the **Editable** menu item.

After entering the Hyperlink's properties in the dialog and pressing **OK**, you will be presented with a "placement cursor". When you click with the cursor, the Hyperlink object is inserted at that location.

You can access the dialog for editing the Hyperlink's properties by selecting **Properties…** from the context menu for the object.

When you double-click on a Hyperlink object, GoldSim will immediately open the application associated with the link (e.g., Internet Explorer, Word).

You can move a Hyperlink object by selecting it and dragging it around the graphics pane. You can resize a hyperlink by selecting the object and dragging one of the "handles" (there are eight handles - one on each corner and one on each side). Note, however, that these operations are only available if you have checked the Hyperlink as being Editable.

Specifying Addresses for the Various Hyperlink Types

The details regarding how the various Hyperlink types are specified are provided below:

Web or local document: To insert a link to a URL which starts with www, you need only type in the address (e.g., www.goldsim.com). To insert a link to a URL which does not start with www, you must specify the transfer protocol (e.g., http://website.com).

To insert a link to some other document (e.g., a Word document or a PowerPoint presentation), you can provide the full path (e.g., C:\Docs\Projects\report.doc), or if the document is in the same directory as the model file, you can provide just the file name. That is, if you only provide a file name, GoldSim will look for the document in the active directory (the directory containing the current model file). You can also use the Browse… button to find a particular file to which you wish to create a hyperlink.

One common type of document that you may wish to open is a Microsoft Excel file (*.xls, *.xlsx or *.xlsm). When opening a spreadsheet file, in some cases you may want to immediately jump to a specified location

(sheet and cell range) when opening the file. GoldSim provides specialized syntax to support this. The syntax is as follows:

Filename#sheetname!cellrange (e.g., Book1.xlsx#Sheet1!a1:c3)

The following points should be noted:

- The entire text can be wrapped in quotes. This should be done if the filename or sheet name contain any spaces.
- For an Excel file, GoldSim will recognize extension .xls, .xlsx, or .xlsm.
- The filename can be specified as a relative or absolute path. Absolute filenames must start with the drive letter or be a UNC path (starting with "\\"). Relative paths can contain "." and ".." as well as folder names (e.g., ".\..\DataFolder\file.xlsx"). Relative filenames with no path are assumed to refer to the folder in which the model file is located.
- # is the delimiter between the filename and the sheet name. It is optional (you do not need to provide a sheet name).The sheet name is not case-sensitive.
- ! is the delimiter between the sheet name and the cell range. It is optional (you do not need to provide a cell range). The cell range can refer to a single cell (e.g., A2) or a range (e.g., A1:A3).

Email to…: When you click on an email link, it opens your default email program with the specified address in the "To" field. You should enter an email address (e.g., support@goldsim.com).

FTP: To insert a link to an ftp site, simply type in the address (e.g., ftp.asite.com).

Execute Command: This allows you to run a specific application. This can be useful, for example, if you wish to open a document with an application which is *not* the default application with which the file extension is associated:

C:\graphics\paint.exe c:\images\bitmap.doc

Note that when using this type of Hyperlink, if the filename has spaces in it, it must be enclosed in quotation marks:

C:\graphics\paint.exe "c:\images\my bitmap.doc"

Go to Element. This causes GoldSim to immediately jump to (but not open) the specified element. You can also specify a full path when using the element command. When you do so, the path name must start with a backslash:

\Container1\Container2\XYZ

Open Container. This causes GoldSim to immediately display the contents of the specified Container. GoldSim searches upward through the containment hierarchy from the current location and jumps to the first Container with the specified name. If you need to be more explicit (i.e., you have multiple Containers with the same name), you can also specify a full path when using the Container command. When you do so, the path name must start with a backslash:

\Container1\Container2\Container3

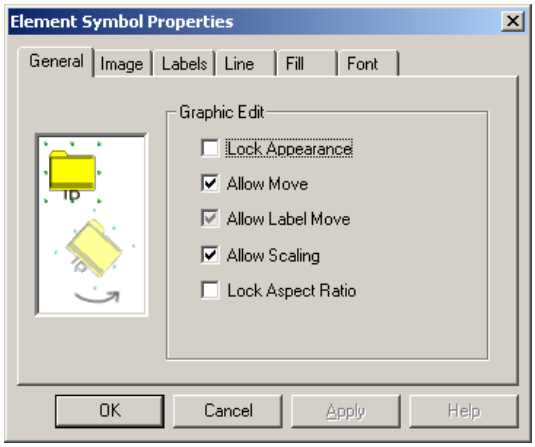
Open Dashboard. This causes GoldSim to immediately jump to the specified Dashboard. You can also specify a full path when using the Dashboard command. When you do so, the path name must start with a backslash:

\Container1\Container2\InputScreen

Note: Dashboards are discussed in detail in the **GoldSim Dashboard Authoring Module User's Guide**.

Changing the Appearance of a Hyperlink Object

You can edit the appearance of a hyperlink by right-clicking on the object and selecting **Appearance…** from the context menu. The following dialog will be displayed.



Note: You can only change the appearance of a Hyperlink object if it is specified as being Editable. Editing can be enabled and disabled by right-clicking on the object and setting or clearing the **Editable** menu item.

The dialog is tabbed. The first tab (**General**) controls the types of changes that can be made to the appearance of the object:

Lock Appearance: If you check this box and then close the dialog, you can no longer move or scale the object in any way. To unlock appearance editing, you must select **Unlock Appearance** from the context menu for the object.

Allow Move: If this box is checked (the default), you can move the object around within the graphics pane. Otherwise, its position remains fixed.

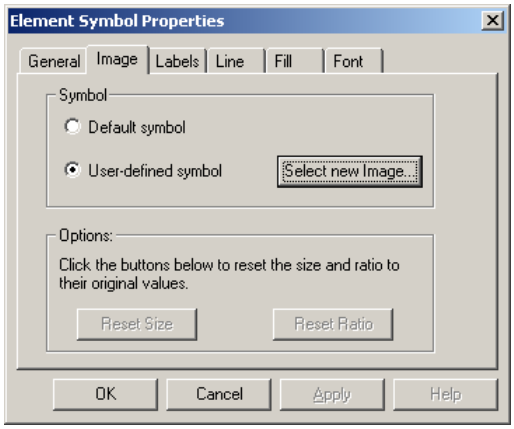
Allow Label Move: If this box is checked you can select and move the object's label (separately from the object itself). If the box is cleared (the default), the label can only be moved with the object itself.

Allow Scaling: If this box is checked (the default), you can change the size of the object by selecting it and dragging one of the control handles (small boxes on the edges of the image). Otherwise, the size is fixed.

Lock Aspect Ratio: If this box is checked (the default), the aspect ratio of the image is locked when it is scaled. Otherwise, you can stretch the image and change the aspect ratio.

Changing the Hyperlink Object's Symbol

The **Image** tab of the Hyperlink Appearance dialog allows you to change the image used for the Hyperlink object in the graphics pane.



If you select the **User-defined symbol** radio button within this tab, you will be prompted for the name of a file to be used for the symbol. The file must be an enhanced Windows metafile (.emf)

Note: GoldSim utilizes enhanced metafiles for symbols because they are vector graphics (and therefore scale without losing image quality). Most advanced graphics programs can create enhanced metafiles (or convert other formats to an enhanced metafile). Note, however, that unless you create the original image as an enhanced metafile, it will not be a true vector graphic and will lose image quality when scaling.

You can use GoldSim's graphic capabilities to convert other graphics formats to an enhanced metafile as follows:

1. Insert a graphic image (e.g., a bitmap) into the graphics pane;
2. Select the image and click **Graphics | Export…** from the main menu (or press **Ctrl+E**);
3. Specify that you wish to save the file as an enhanced metafile.

Read more: Adding Images (page 707).

Once the symbol is defined using a user-defined symbol, you can change the symbol by pressing **Select new Image…**, or you can switch back to the default image provided by GoldSim by selecting the **Default symbol** button.

You can access the dialog for selecting a new image directly (without using the Appearance dialog) by selecting **Change Symbol…** from the context menu for the element.

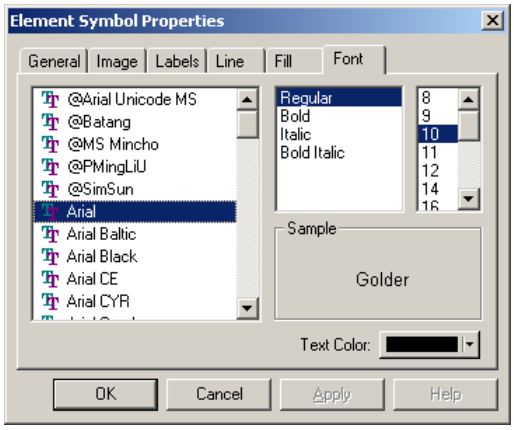
Two additional buttons are also on the **Image** tab:

Reset Size: This resets the size of a scaled image back to its original size.

Reset Ratio: This resets the aspect ratio of a scaled image back to its original ratio (by changing the symbol's height).

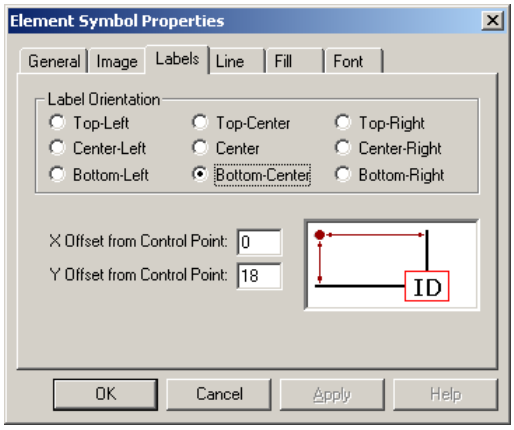
Changing the Hyperlink Object's Label

Two tabs within the Hyperlink Appearance dialog control the appearance of the object's label. The **Font** tab controls the label's font.



The options in this dialog are self-explanatory.

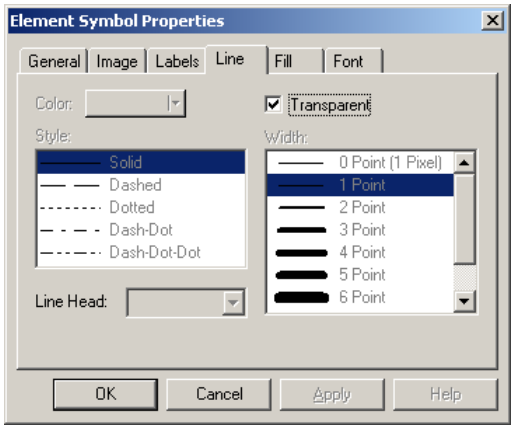
The **Labels** tab controls the position of the label relative to the symbol.



You can choose one of nine positions for the label. If one of these nine positions is not sufficient, you can manually move the label by selecting just the label and dragging it. However, in order to manually move the label in this way, the **Allow Label Move** box must be checked in the **General** Tab.

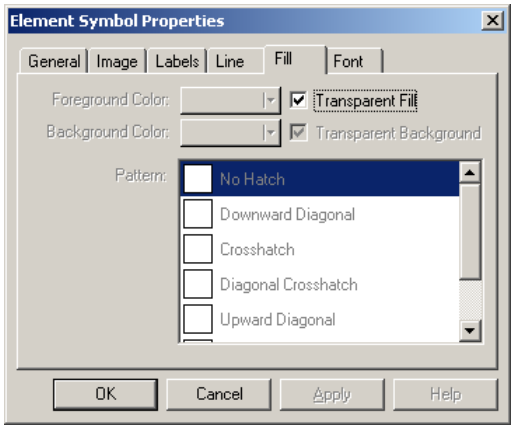
Changing the Hyperlink Object's Background and Outline

Two tabs within the Hyperlink Appearance dialog control the appearance of the object's background and outline. The **Line** tab allows you to draw a line (a box) around the symbol and label.



By default, the line is transparent. If you clear this box, you can change the line's color, style and width. Note, however, that the style can only be applied to 0 point lines.

The **Fill** tab allows you to change the appearance of the fill for the symbol and label.



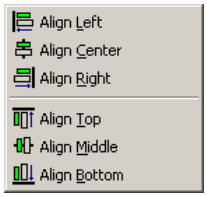
The fill consists of a background and a foreground. The **Foreground Color** is superimposed on top of a **Background Color** according to a selected **Pattern**. By default, both are transparent.

Manipulating Graphical Objects

GoldSim provides a number of utilities for aligning, ordering, spacing, sizing, grouping, rotating, pasting and moving graphical objects (including element symbols) in the graphics pane.

Aligning and Ordering Objects

If you have selected multiple objects and/or elements in the graphics pane, you can choose **Graphics|Align** from the main menu to access a submenu.



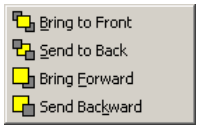
The options in this menu (which are self-explanatory) can be used to align the selected objects.

When you add an object to the graphics pane, it is assigned a particular order. The order determines how objects are "stacked" (i.e., whether an object is in front or behind another object) when they overlap. Higher order objects are always placed in front of lower order objects:

With regard to order, GoldSim groups objects into three classes: 1) elements; 2) influences; and 3) everything else (graphical objects like images and text). Elements always have a higher order than influences and influences always have a higher order than graphical objects. Hence, influences will always appear behind elements, and images and text will always appear behind influences.

Within a given class of objects (i.e., elements, influences or graphical objects), the latest object added to the graphics pane always has the highest order. Hence, if you add three circles, the first one added will have the lowest order and the last one added will have the highest order.

Once an object is placed in the graphics pane, you can manually change its order (within its class of objects) by selecting the object and choosing Order from the context menu to display a submenu.



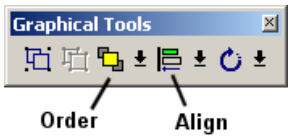
Bring to Front brings the object all the way to the front (gives it the highest order).

Send to Back sends the object all the way to the back (gives it the lowest order).

Bring Forward increases the object's order by one.

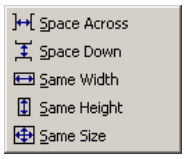
Send Backward decreases the object's order by one.

The order and align options are also available in the Graphical Tools toolbar:

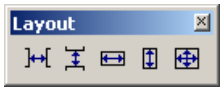


Spacing and Sizing Objects

If multiple objects are selected, you can also choose to space them evenly (horizontally or vertically), or set their widths, heights or sizes to be identical. These options are provided under **Graphics|Layout** in the main menu:



The layout options are also available from the Layout toolbar:



Precisely Moving Objects

Although you can move objects around the graphics pane by simply dragging them from one location to another, in some cases you may want to move them in a more precise manner. To facilitate this, GoldSim allows you to move objects vertically and horizontally using the arrow keys.

There are two modes of movement using the arrow keys:

- Using the arrow keys alone moves the object a short distance in a horizontal or vertical direction.
- Holding the **Shift** key down while using the arrows increases the distance moved by a factor of five.

Note also that when moving objects in the graphics pane, you can force their upper left hand corner to snap to a grid.

Read more: The Graphics Pane Grid and Background (page 386).

Grouping Objects

If you have selected multiple objects in the graphics pane, you can choose **Grouping|Group** from the context menu of one of the objects or **Graphics|Group|Group** from the main menu to group the objects. When objects are grouped, they behave as a single object. Hence, when you select one object, all objects in the group are selected. When you edit the graphical

property (e.g., font, background color) of one object, the corresponding property of all objects in the group is modified.

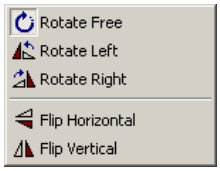
Note: Elements cannot be a member of a group. Only graphical objects, text, images and hyperlinks can be members of a group.

To Ungroup the set of objects, select the Group, and choose **Grouping|Ungroup** from the context menu of one of the objects or **Graphics|Group|Ungroup** from the main menu.

The Group and Ungroup options are also available in the Graphical Tools toolbar:

Rotating Objects

You can rotate a graphic object or text object in the graphics pane by selecting the object and choosing **Graphics|Rotate** from the main menu to access a submenu.



Rotate Left rotates the object 90 degrees counter clockwise.

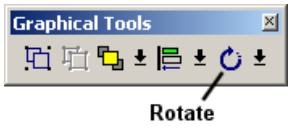
Rotate Right rotates the object 90 degrees clockwise.

Flip Horizontal flips the object horizontally (around a vertical center axis).

Flip Vertical flips the object vertically (around a horizontal center axis).

Free Rotate activates a Rotate Cursor (when placed over the object). Dragging the cursor within the object rotates it about its center point.

The Rotate options are also available in the Graphical Tools toolbar:



Copying, Pasting, Moving and Deleting Graphical Objects

You can copy and paste objects within the graphics pane using **Ctrl+C** and **Ctrl+V**, respectively.

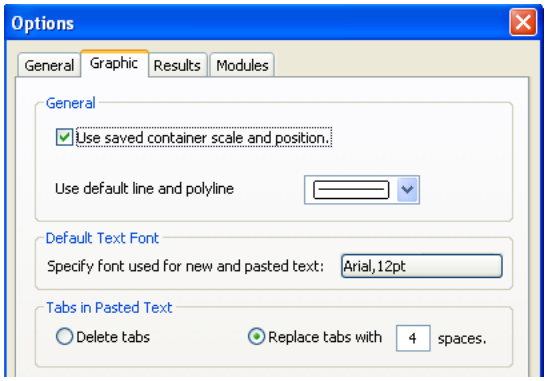
You can delete an object by selecting it and pressing the **Delete** key.

You can also use the context menu for the object to **Cut**, **Copy**, **Paste** or **Delete**.

You can paste graphical objects and text from one GoldSim model to another and/or from other applications into GoldSim. For example, you could paste a bitmap from a graphics program directly into the GoldSim graphics pane, or you could paste text from a word processing application or text editor directly into the graphics pane.

When you paste text from another application into the GoldSim graphics pane, you must tell GoldSim how it is to treat any tabs that are present in the pasted

text, and the font for the text. This is specified in the **Graphic** tab of the Options dialog (accessed via **Model|Options…** from the main menu).



You can also move graphical objects from one container to another in the same manner that you can move elements.

Read more: Moving Elements Between Containers (page 106).

Using the Graphical Undo and Redo Functions

GoldSim provides an automated "undo" and "redo" capability for actions involving the manipulation of graphics. These are accessible from the main menu (**Edit|Undo** and **Edit|Redo**). Alternatively, you can use **Ctrl+Z** and **Ctrl+Y** for Undo and Redo, respectively.

The Undo function only operates on actions involving the manipulation of graphic objects, such as adding, moving or modifying graphics and text. It cannot Undo actions that affect the model itself (such as editing inputs to an element).

The "stack" of actions that can be undone is terminated as soon as you carry out an action which cannot be undone. For example, if you add a text object, then add a rectangle, and then edit its graphical properties, you can step backward and undo these actions. If you subsequently added a new element, however, the "stack" of graphical actions would be deleted (and could not be undone using the Undo function).

The Redo function is only available after you Undo an action. That is, its function is to reverse an Undo.

Creating Printed Documentation

GoldSim provides two mechanisms to create printed documentation of your model:

- You can print the graphics pane; and
- You can export the graphics pane (or one or more selected objects) to a graphics file.

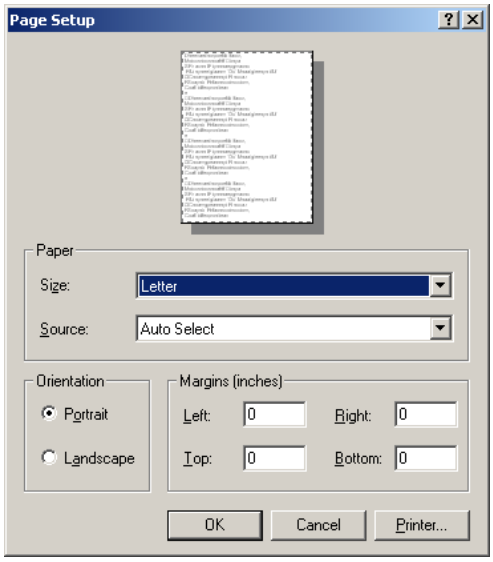
Printing the Graphics Pane

If you scroll far enough horizontally or vertically you will eventually reach the "edge" of the graphics pane. That is, the graphics pane represents a document of fixed size. This means that an element on the screen scales to a specific size on paper, and the entire graphics pane maps to a printed document with specific dimensions.

Read more: Adjusting the Size of the Graphics Pane (page 388).

In order to print the document, you must specify the manner in which the document is to be represented on paper. In particular, you must specify the paper

size and orientation, along with the margins. You do this from a dialog accessed **File|Page Setup…** on the main menu:



When you make a change to the Page Setup (and press OK), GoldSim will ask if you want to apply these settings to all Containers, or just the current Container.

You can view the Page Bounds within the graphics pane by selecting **Properties…** from the context menu for the graphics pane (and then checking **Show Printer Page Boundaries** in the Graphics tab that is displayed.


***Print Preview** button*

You can preview the printed document by selecting **File|Print Preview** from the main menu or pressing the **Print Preview** button in the Standard toolbar.


***Print** button*

You can print the graphics pane directly from the Print Preview window, by selecting **File|Print** from the main menu or pressing the **Print** button in the Standard toolbar.

Exporting the Graphics Pane

You can export the graphics pane and save it to a graphics format.

The simplest way to do this is to use **Alt+PrtSc** to cut the entire window to the clipboard (as a bitmap).

You can also use GoldSim's Export utility to save just the graphics pane (or selected items in the graphics pane) to a particular graphics format.

To do so:

1. Select an object (or objects) in the graphics pane, and select **Graphics | Export…** from the main menu (or press **Ctrl+E**). A dialog for specifying a file type and a location for the file is displayed.
2. Select a file type and a location, and then press **Save**.

If no object is selected in the graphics pane when you choose **Graphics | Export…** from the main menu (or press **Ctrl+E**), the entire graphics pane will be exported.

Creating a GoldSim Player File

The GoldSim Player is a special version of GoldSim that allows you to "play" or “read” an existing GoldSim model without having to license the GoldSim software. In general, the user interface for the GoldSim Player is identical to that

of the full GoldSim version, with a number of menu options and controls for editing the model removed or disabled.

The GoldSim Player can be downloaded (for free) from the GoldSim website. It is also automatically installed on your machine when you install GoldSim (and is available via the Windows Start menu).

The GoldSim Player can not read a normal GoldSim file (a .gsm file). Rather, the GoldSim Player can only read Player files (.gsp). You can save a copy of a GoldSim model as a Player file by selecting **File|Save Player File…** from the main menu.

When you do so, a wizard will appear to assist you in creating the Player file. The first page of the wizard simply provides some explanatory text. The second page of the wizard prompts you for the Author's Name, a Model Description, and a Model Title. The Author's Name, Model Title, and Model Description can be viewed via a menu item in the Player (and the Model Title also appears in the title bar of any Dashboards you create). The third page of the wizard allows you to specify the properties of the Player file:

If the **Allow to browse the model** option is checked, the Player user is allowed to leave any Dashboard(s) and browse the structure of the model. Note that if the model does not have at least one Dashboard and/or the Dashboard includes a Button control that jumps to a Container or an element, the model is automatically browsable and this option will be checked on and grayed out.

If the **Allow the user to run the model** option is checked, the Run Controller within the Player will be displayed and the user can use this to run (and save) the model. If this option is off, the Run Controller is not displayed and the Player user will be able to view the model, but will not be able to run (or save) it. If the user can run the model, you can also choose whether to **Allow changes to the Time Settings** and **Allow changes to the Monte Carlo settings**. These allow the Simulation Settings of the model to be edited within the Player.

The final field in the third page of the wizard determines what will be displayed when the Player user opens the file. If the Player file has no Dashboards, the only option is "Top-level Container". However, if the file contains one or more Dashboards, you can specify the ***Default Dashboard*** ; that is, the Dashboard that will be displayed when the file opens in the Player. The drop-list contains all Dashboards in the model. If the model is browsable, it also contains "Top-level Container".

The final page of the wizard is used to specify the path and filename of the Player file.

Player files can be saved with or without results. If you try to create a Player file while you are in Edit Mode, GoldSim will first check the model for errors. If it finds errors, it will not allow you to create the Player file.

A Player file cannot be read by GoldSim; it can only be read by the GoldSim Player. You can view and navigate a Player file using the GoldSim Player just as you would view and navigate a GoldSim model file in GoldSim. This includes the ability to open most GoldSim dialogs (e.g., in order to examine how an element has been defined).

Note: If the model file from which the Player file is created was password-protected, the Player file will also be password-protected, and the user will need to enter a password to open the file.

Read more: Password-Protecting a Model File (page 75).

As pointed out above, the user interface for the GoldSim Player is identical to that of the full GoldSim version, with a number of menu options and controls for editing the model removed. Instructions for using the GoldSim Player are described within the program's online help system, as well as the **GoldSim Player User's Guide**.

Creating a Player File Using the Dashboard Authoring Module

The GoldSim Dashboard Authoring Module provides tools that allow a GoldSim modeler to design and construct a "dashboard" interface for models.

The interfaces can be designed to look like "dashboards" or "control panels", with buttons, gauges, sliders and display panels, and the author can embed instructions, tool-tips and graphics to provide instructions on the use of the model.

Such an interface allows a model to be easily used by someone without requiring them to be familiar with either the GoldSim modeling environment or the details of the specific model.

When such "dashboarded" models are saved as Player files, Player users can not only view and navigate the model, but they can also modify the model (through the dashboards), save the changes, and run the model.

The Dashboard Authoring Module is described in detail in the **GoldSim Dashboard Authoring Module Users Guide**.

Chapter 10: Advanced Modeling Concepts

These are much deeper waters than I had thought.

Sherlock Holmes (Sir Arthur Conan Doyle),
The Reigate Squires

Chapter Overview

This chapter describes a number of advanced and powerful GoldSim features. They have been saved for last since in order to use them, you must have a good understanding of the basic GoldSim features. Although you can build many kinds of models without using these features, once you become comfortable with GoldSim, many of these advanced features will seem indispensable, and you will want to take advantage of some of them in all of your models.

In this Chapter

The following topics are discussed in this chapter:

- Using Vectors and Matrices
- Understanding Locally Available Properties
- Solving Convolution Integrals
- Generating Stochastic Time Histories
- Using Resources
- Script Elements
- Using Conditional Containers
- Using External Application Elements
- Localizing Containers
- Cloning Elements
- Referencing an Output's Previous Value
- Using Looping Containers
- Understanding the Causality Sequence
- Using SubModels to Embed Models Within Models
- Customized Importance Sampling Using User-Defined Realization Weights
- Dynamically Revising Distributions Using Simulated Bayesian Updating
- Tracking Model Changes

- Linking Elements to a Database
- References

Using Vectors and Matrices

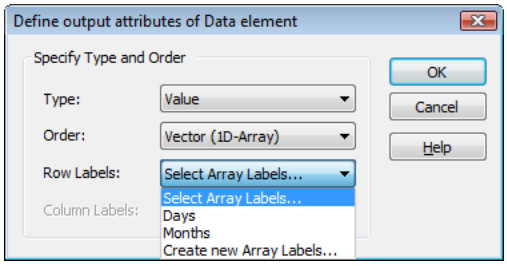
In many systems, you will want to create and manipulate elements that represent collections of data, rather than individual items. For example, you may want to create an element that represents your company's revenue for each of the last five years, or an element that represents the salmon population in each of 20 streams.

One way to do this, of course, would be to create separate elements for each object you wanted to model (e.g., five elements representing revenue, 20 elements representing salmon populations).

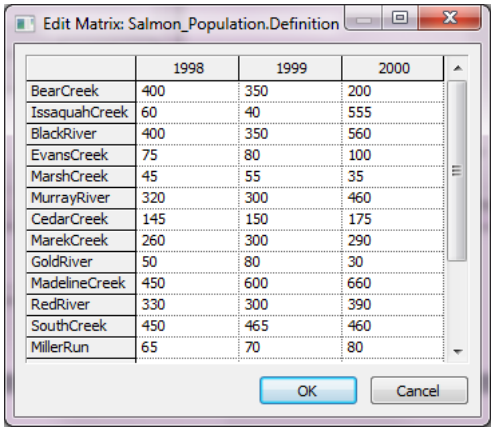
Such an approach, however, is not desirable for two reasons:

- It could require you to create a very large number of elements (e.g., if you wanted revenues for the past 25 years or wanted to evaluate salmon populations in 200 streams). This could result in very large, cluttered models.
- Usually, you will want to carry out the same types of calculations and operations on all related objects in such a collection (e.g., multiply all revenues by 2, compute the number of salmon eggs this year for all 20 streams based on the current salmon population). Having to do this individually for every object in a large collection would be very cumbersome and time-consuming.

To address these kinds of problems, GoldSim allows you to create and manipulate vectors and matrices (collectively referred to as arrays). For example, you could create a vector element that represented the salmon populations in each of 20 streams:



You could also create a matrix element that represented the salmon population in each of 20 streams over a period of three years:



In addition to adding data in the form of vectors and matrices, you can manipulate these arrays in equations. For example, you could create an Expression element, and define it as:

2 * Salmon_Population

The output of the Expression would be an array, identical to the Salmon_Population array, except each item of the array would be two times greater.

If you did not use arrays to carry out this calculation, rather than creating 2 elements, you would need to create 40 elements (20 Data elements and 20 Expression elements) to accomplish the same thing!

Vectors and matrices are applicable to almost any kind of system, and you will likely want to take advantage of this feature in most of your models.

An example model which uses vectors and matrices within different elements (Arrays.gsm) can be found in the General Examples folder in your GoldSim directory. In addition, TimeSeries_Array.gsm (in the Time Series subfolder of General Examples) includes examples of how Time Series elements can be used to generate arrays.

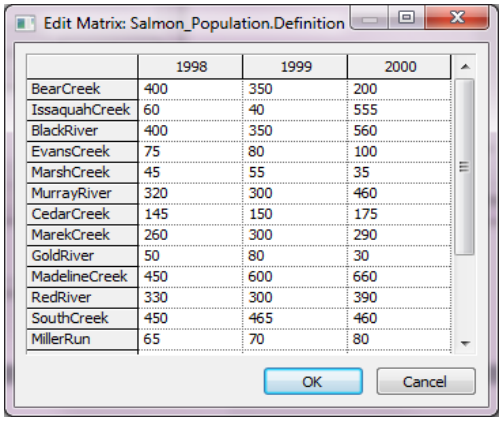
The following sections describe how you can create and use vectors and matrices in your models.

Understanding Array Labels

When you define a vector or a matrix, it must be defined relative to a particular set of array labels. A set of array labels is simply a collection of labels for the items of the array. The labels can be numbers (1, 2, …, 10) or words (apples, oranges, peaches, pears). When you define an element which has an array as an output, you must specify the sets of array labels upon which the array is based.

You define and reference a particular item of an array by using these labels. Vectors require a single set of array labels, and matrices require two sets of array labels (one for the rows, one for the columns).

As an example, consider a matrix that represents the salmon population in each of 20 streams over a period of 3 years:



The matrix is defined relative to two sets of array labels: the rows are labeled using a "stream" set, and the columns are labeled using a "year" set. The labels or members for the "stream" set are BearCreek, IssaquahCreek, BlackRiver, and so on. The labels for the "year" set are the numbers 1998, 1999, and 2000.

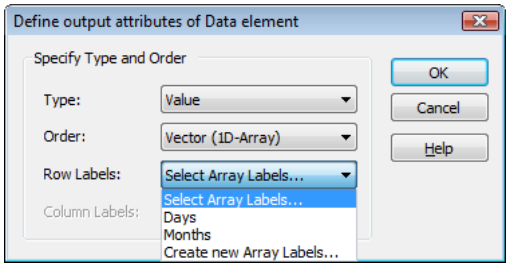
You would reference the (scalar) item in the third row and second column of the matrix as Salmon_Population[BlackRiver, 1999].

Creating and Editing Array Labels

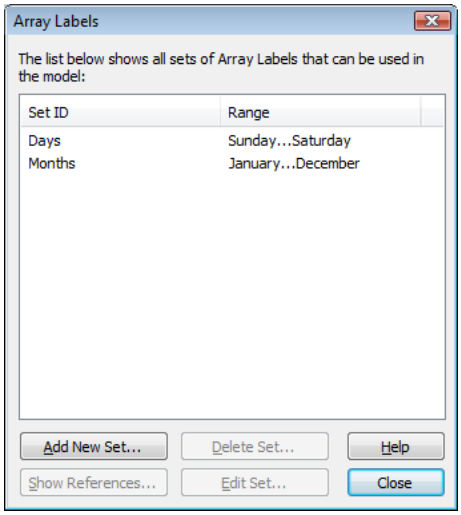
You can create (and edit) sets of array labels by selecting **Model|Array Labels…** from the main menu.

You can also create array label sets directly from the dialog for defining the Output Attributes for an element.

Read more: Defining Vectors and Matrices Using Data Elements (page 733).

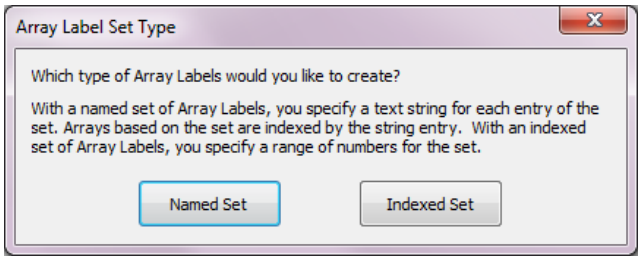


In either case, the following dialog for defining the array labels is displayed:

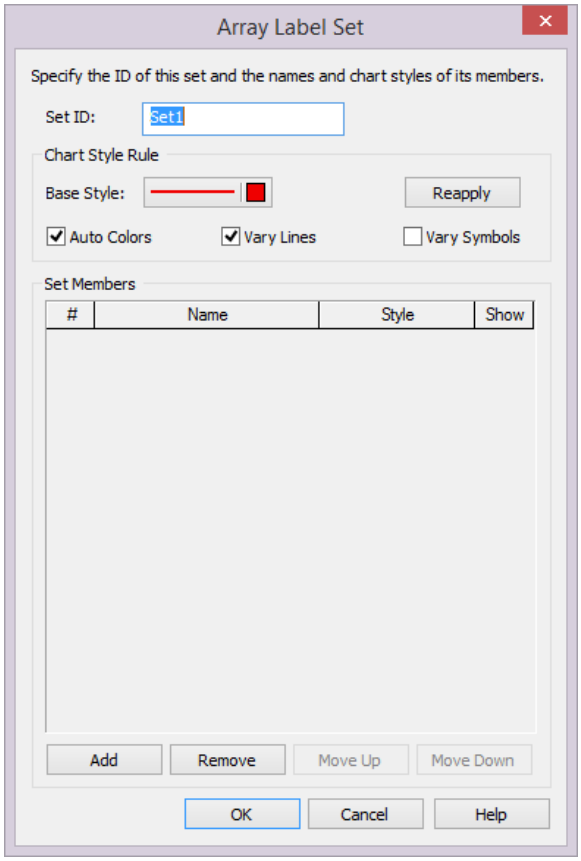


For convenience, GoldSim provides two commonly-used sets of array labels: Days has seven items (the days of the week); and Months has twelve items (the months of the year). GoldSim extension modules may also add some special sets (e.g., the Contaminant Transport Module adds a Species and an Elements set).

To edit an existing set of array labels, select the set from the list in the Array Labels dialog and press **Edit Set…**. To add a new set of array labels, press **Add New Set…**. If you are adding a new set, a dialog for specifying the type of array labels is displayed:



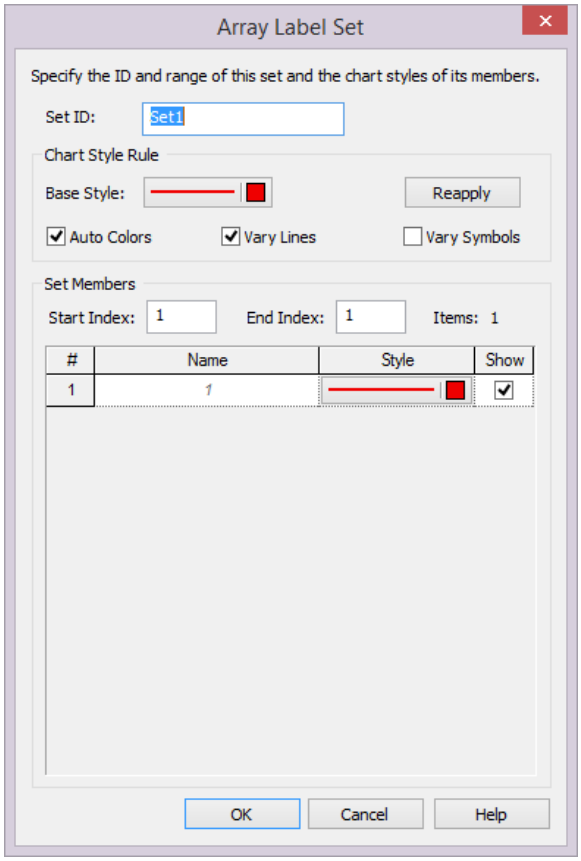
You must select whether you wish to define a Named Set or an Indexed (numbered) Set. If you choose to create a Named Set of array labels (or select an existing Named set to be edited), a dialog for editing the set will be displayed:



You can assign a name to a new set of array labels (by default it will be named Set*n*). You add and remove entries using the **Add** and **Remove** buttons. Holding the **Ctrl** key down changes the **Remove** button to a **Remove All** button, so you delete all entries. You can also paste a column of labels (e.g., from a spreadsheet) within this dialog by pressing **Ctrl+V**.

After you add two or more entries, the **Move Up** and **Move Down** buttons become available, allowing you to move entries up or down in the list. If you reorder the items, any items in existing elements using the set will automatically be re-sorted to the new sequence.

If you choose to create an Indexed set of array labels (or select an existing Indexed set to be edited) a dialog for editing the set will be displayed.



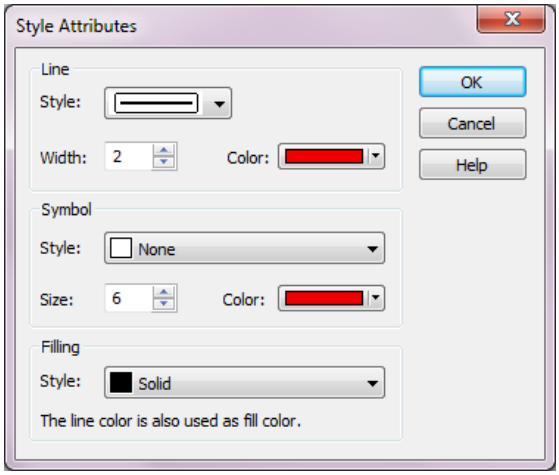
You can assign a name to the new set of array labels (by default it will be named Set*n*).

You must then enter a **Start index** (e.g., 1) and an **End index** (e.g., 100). When you press the **Reapply** button, GoldSim will then create all the entries for the set.

If you are editing an existing indexed set which is already referenced by some elements for which data has been defined, and change the Start index, GoldSim will prompt you for how you would like to adjust the values in existing Data elements that reference that set.

Read more: Changing the Start Index for an Indexed Set that is Already Being Referenced (page 731).

The Array Label editing dialog also allows you to define the Styles that are used when an element based on the set is displayed in a result chart. The **Base Style**, in combination with the **Auto Colors**, **Vary Lines** and **Vary Symbols** options allow you to automatically generate the Styles by pressing the **Reapply** button. Alternatively, you can click on any Style in the list to manually specify the Style attributes:



Note: The Array Label Set dialog can also be accessed when viewing results (in order to access the Chart Style attributes). When accessed in this manner, in some cases, the Chart Style options may be hidden (if editing them is not applicable).

The **Show** checkbox determines whether or not the item is displayed in result charts.

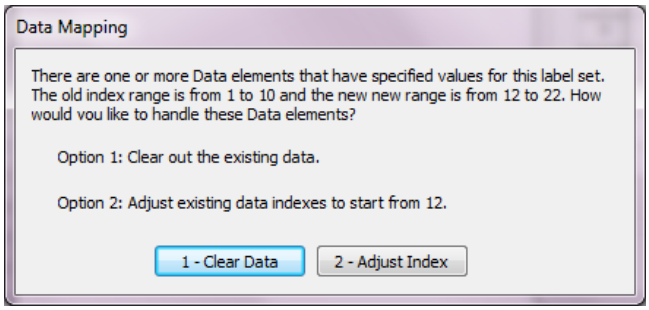
Read more: Viewing Time Histories for Array Outputs (page 547); Viewing a Vector Chart (page 651); Viewing a Matrix Chart (page 653).

Note: Array label sets are saved with your model. If you wish to use the same set of array labels in different models, you must recreate it in each model. Alternatively, you can copy a set of array labels from one model file to another by copying an element that uses the set. Note, however, that GoldSim will only allow you to do so if a set of the same name does not exist in the model file into which the element is being copied.

Changing the Start Index for an Indexed Set that is Already Being Referenced

If you are editing an existing indexed set which is already referenced by some elements for which data has been defined, and change the Start index, GoldSim will prompt you for how you would like to adjust the values in existing Data elements that reference that set.

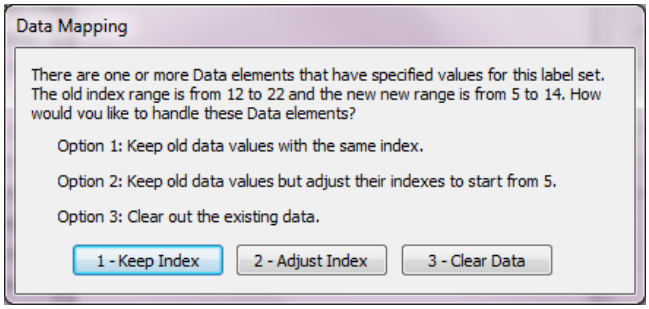
In particular, if the new range does not overlap with the old range, a dialog like this will appear to allow you to specify how the data is to be mapped to the new index values:



If you select Option 1, all data in existing Data elements will be cleared out (set to 0).

If you select Option 2, existing data will be mapped such that any data associated with the old Start Index will be mapped to the new Start Index, and so on. If the new range is shorter, data at the end of the range will be lost. If the new range is longer, zeros will be added to the new items.

If the new range does overlap with the old range, a slightly different dialog will be displayed for mapping the data:



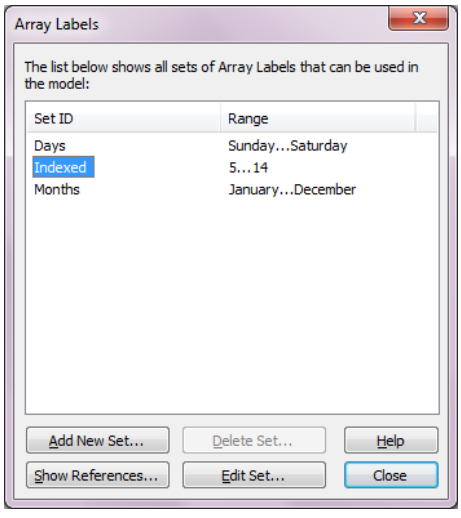
If Option 1 is selected, GoldSim will only map entries for items in the old set which match items in the new set. For example, if the original set was 1 through 20, and the new (edited) set was 15 through 34, for any elements that referenced the set, only entries 15 through 20 would be mapped to the new set (as the first five entries), and entries associated with items 1 through 14 in the original set would be lost. The remaining entries associated with the new set (16 through 34) would all be set to zero.

If you select Option 2, existing data will be mapped such that any data associated with the old Start Index will be mapped to the new Start Index, and so on. If the new range is shorter, data at the end of the range will be lost. If the new range is longer, zeros will be added to the new items.

If you select Option 3, all data in existing Data elements will be cleared out (set to 0).

Deleting a Set of Array Labels

You can delete a set of array labels by selecting the set in the Array Labels dialog and pressing **Delete Set…**.



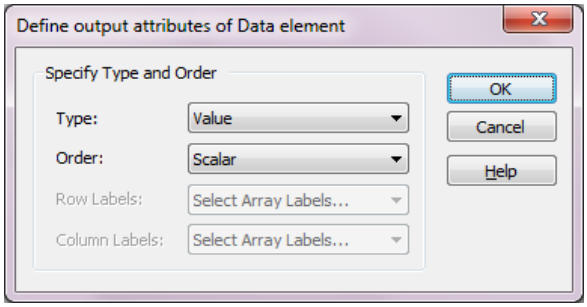
Note, however, that this button will be grayed out if the set is being referenced somewhere in the model (you can not delete a set if it is referenced by an

element). If a set is used in the model, the **Show References…** button will be available. Pressing this button displays a dialog that provides a list of all of the elements which reference the set, and allows you to edit or delete each element.

Defining Vectors and Matrices Using Data Elements

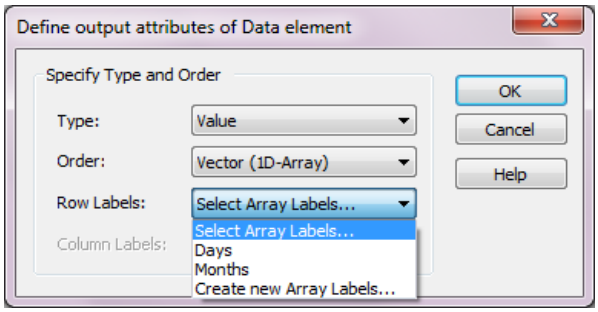
Once you have defined the sets of array labels that you wish to use, you can create vectors and matrices. The primary way to create vectors and matrices is to define them using Data elements. Once these exist, other vectors and matrices can be created by other kinds of elements based on these (e.g., by using these as inputs).

You specify that a Data element is to be a vector or a matrix when you define its output attributes (by pressing the **Type…** button in the element's properties dialog). The dialog for specifying output attributes for an element looks like this:

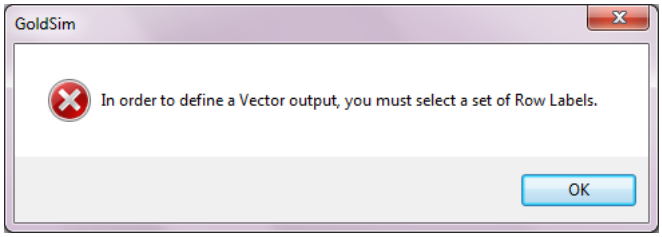


The **Order** drop-list provides three choices: "Scalar" (the default), "Vector (1D-Array)", and Matrix (2D-Array)".

If you wish to create a vector, select "Vector (1D-Array)". If you wish to create a matrix, select "Matrix (2D-Array)". If you select "Vector (1D-Array)", the **Row Labels** field becomes available. If you select "Matrix (2D-Array)", both the **Row Labels** and **Column Labels** fields become available. These two fields contain lists of all of the sets of array labels defined in your model:



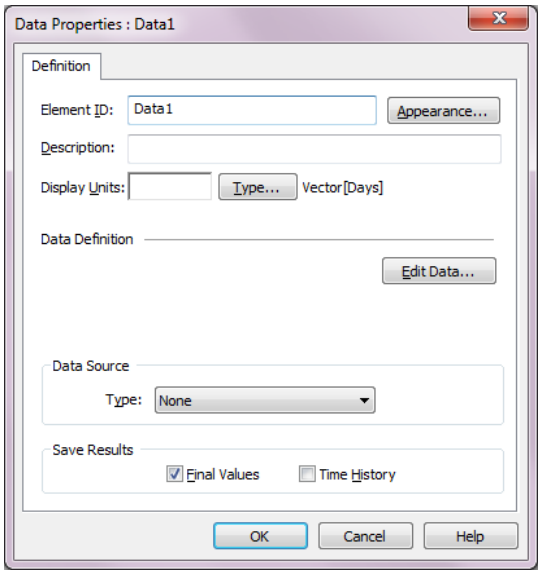
If you have defined the Order as a vector or a matrix (i.e., if the Row Labels or Column Labels fields are active), you must select a set of array labels (or GoldSim will display a warning message when you try to close the Output Attributes dialog).



Note that within the drop-list for the Row and Column labels is the choice to "Create new Array Labels". If you click this option, the dialog for defining new array labels will be displayed. After leaving the Array Labels dialog, you will be returned to the dialog for defining output attributes.

Read more: Creating and Editing Array Labels (page 728).

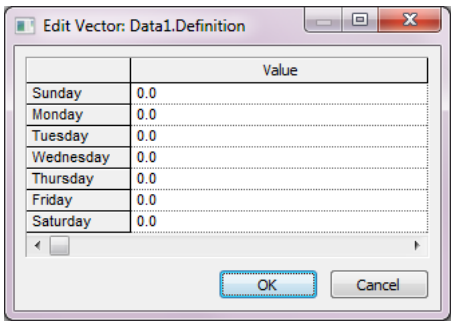
To illustrate how you would create a vector or a matrix Data element, consider the following example. Suppose that you defined the **Order** as "Vector (1D-Array)", and defined the **Row Labels** as "Days". The properties dialog for the Data element would look like this.



Note that this dialog is different from the standard Data element dialog for a scalar.

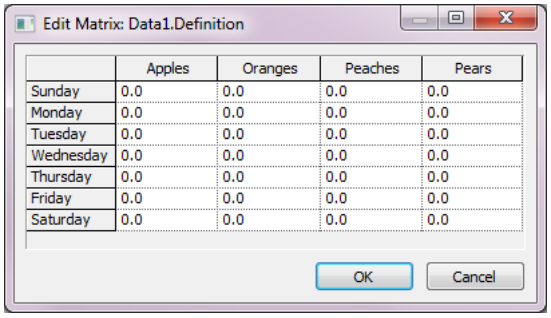
Read more: Data Elements (page 154).

In particular, because it is an array, you cannot directly edit the Data input field. To edit the vector, you must press **Edit Vector…**, which will display the following dialog.



This dialog displays all items of the vector (in this case, seven). Each field is a standard edit field (you can enter a number, an expression or a link), although typically you would enter just a number (with units).

Note that the equivalent dialog for a matrix looks like this (in this example using a set of array labels for the Column Labels named Fruit with members Apples, Oranges, Peaches and Pears):



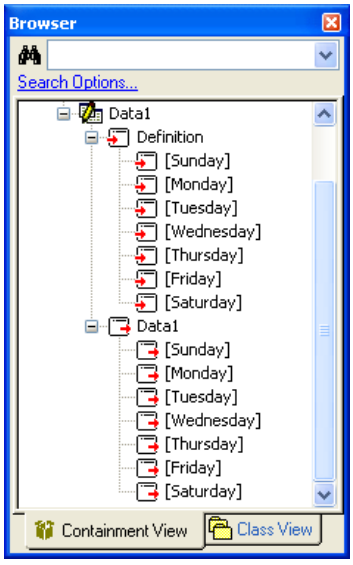
Although you would typically enter constants (numbers) into the input fields defining a vector or a matrix, the fields accept links and expressions. For example, you could create a vector of Stochastics by entering a link to a Stochastic for each item in a vector.

Note: You can paste data from a spreadsheet, a Word table, or a tab-delimited text file directly into a vector or matrix grid. To do so, copy the data to the clipboard, select the cell in the vector or matrix representing the upper left-hand corner of the range into which you wish to paste (by single-clicking in it), and press Ctrl+V.

Note: You can resize the columns of the editing dialog by dragging the lines separating columns right or left. You can also resize the entire dialog by dragging a side (or a corner).

Viewing an Array in a Browser or Interface

When you view a vector or a matrix in a browser or an interface, the inputs and outputs can be expanded to show the individual items.



If you double-click On the Minus sign (-) next to a group label with the link cursor, the entire vector is used. If you double-click an individual item with the link cursor, only that (scalar) item of the vector is used.

Note: Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

Note that placing the cursor over an item of the array displays the Current Value (if in Edit Mode) or the Last Value (if in Result Mode). The Current Value is computed as the expected value of the item. The Last Value is the final value (i.e., the value at the end) of the last realization.

Read more: Understanding Result Mode (page 512).

Referencing an Item of an Array

If required, you can access a particular item of the array in an expression. You do this by using brackets [ ].

For example,

Data1[Monday]

references a single (scalar) value representing the second item of the vector Data1 (defined by the set "Days", which begins with Sunday).

Similarly,

Data2[Monday, Peaches]

references a single (scalar) value representing the item in the second row and third column of the matrix Data2 (with the rows defined by the set "Days", and the columns defined by the set "Fruit", which consists of Apples, Oranges and Peaches).

When referencing array items, you can also use variables as the arguments. For example, if Data1 was a vector of "Days", and X was a scalar value, you could write the following in an expression:

Data1[X]

If X was 3, it would return the 3rd item in the vector (i.e., Data1[Tuesday]).

GoldSim rounds to the nearest integer when evaluating an array argument, so in this example, as long as the rounded value of X was between 1 and 7 inclusive, GoldSim would be able to evaluate the expression. If X evaluated to a number outside of that range, GoldSim would display a fatal error. Of course, X could change with time (or any other variable in your model).

This functionality can also be used with matrices. As an example, if Data2 was a matrix with the rows defined by the set "Days", and the columns defined by the set "Fruit" (with 3 entries), you could reference the following in an expression:

Data2[X, Y]

In this case, X would need to be an integer between 1 and 7, inclusive, and Y would need to be an integer between 1 and 3 inclusive. If X was 3 and Y was 2, it would return the item in row 3, column 2 of the matrix (i.e., Data2[Tuesday, Peaches] assuming Peaches was the second array label for "Fruit").

When using variables to reference array items, the following points should be noted:

- If the array is based on a named set, the variable represents the ordinal item in the set (i.e., the row or column number). For example, if Data1 was vector of "Days", and X was 3, then Data1[X] would return the 3rd item in the vector (i.e., Data1[Tuesday]).

- If the array is based on an indexed set, the variable represents the actual item with that number label in the set. For examples, if Data9 was a vector defined by an indexed set with labels 3, 4, 5 and 6, and X was 3, then Data9[X] would return the first item in the vector (i.e., Data9[3]).

Finally, when using this with matrices, you can use * as a wildcard. Using the example above,

Data2[*, Y]

would return column Y (a vector of Days). Similarly,

Data2[X, *]

would return row X (a vector of Fruit).

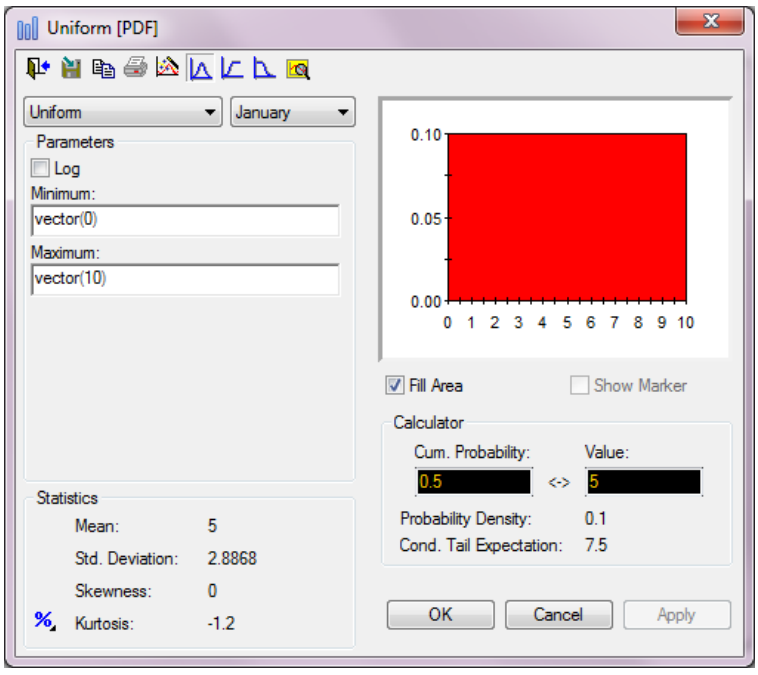
This ability to use variables when referencing array items is particularly powerful when using this feature in conjunction with array constructor functions.

Read more: Defining Arrays in an Input Field Using Array Constructor Functions (page 738).

Defining Vectors Using Stochastic Elements

In addition to using Data elements to create vectors, you can also create vectors using Stochastic elements. If you define a Stochastic element as a vector, rather than inputting a single probability distribution, you specify a set of probability distributions (one for each item of the vector).

If you specify the Stochastic as a vector (by specifying a set of array labels via the **Type…** button), the dialog for defining the distribution (accessed via the **Edit Distribution…** button) will look similar to this:



All the inputs to the Stochastic must be vectors.

Read more: Creating a Stochastic Vector (page 184).

Defining Arrays in an Input Field Using Array Constructor Functions

One of the quickest ways to create an array is to use array constructor functions. These are functions that can be entered into an input field that generate an array (a vector or a matrix) by specifying the array label set(s) and values as arguments.

For example, the following expression generates a vector based on the Days array label set, in which all the items have a value of 1m:

Vector(days, 1m)

The following expression generates a matrix based on the Days array label set (for rows) and the Months array label set (for columns), in which all the items have a value of 0 g:

Matrix(days, months, 0g)

Note: You do not have to specify the array label set(s) when using an array constructor. If you omit them, GoldSim will assume that the constructed array has the same order and set(s) as that of the input field where it is being defined. For example, if you created an Expression defined as a vector with array label set "days", then to create a vector of zeros, you could enter either "Vector(days, 0)" or simply "Vector(0)".

In addition to creating an array in which all the values for the items are identical (and hence specified by a single argument), you can also specify each item separately:

Vector(days, 1m, 3m, 0m, 5m, 6m, 2m, 7m)

In this example, since the Days array label set was specified, seven items are defined.

Note: When defining a matrix using an array constructor, the first argument represents the row array label set (having n items), the second argument is the column label set (having m items), and if all of the values are specified separately, they are defined as follows: r_1c_1, r_1c_2,…r_1c_m, r_2c_1,…r_nc_m. That is, the items are listed left-to-right-then-down through the matrix.

The units for the values of a constructor do not need to be identical. The following expression is valid:

Vector(days, 1m, 3ft, 0km, 5mm, 6m, 2ft, 7m)

If any of the values listed do not have units, they take on the units of the first item in the array:

Vector(days, 1m, 3ft, 0km, 5mm, 6m, 2, 7m)

In the example above, the sixth item of the vector would be 2m.

Vector(days, 1m3/day, 3, 0, 5, 6, 2, 7)

In the example above, all items have units of m3/day.

The values for the items of an array can be specified as links. For example, if X was an Expression with dimensions of volume/time, it could be inserted into the example above:

Vector(days, X)

In this case all 7 items would take on the value of X. Note that links such as X do not have to represent constants. They can be functions of time.

In addition, when using the Matrix constructor, you can specify vector arguments. That is, you can use the Matrix constructor to build a matrix from specified vectors. The rules for doing so are as follows:

- If you create a matrix that has a different number of rows than columns, you can define the matrix by providing vectors for either all of the rows, or all of the columns. GoldSim then intelligently creates the matrix in the appropriate way (i.e., it automatically determines based on the array label sets whether the vectors should be treated as rows or columns).
- If the matrix is square and the array label set of the provided vector matches the column array label set of the matrix, the vector values are loaded into the matrix by row. Under all other circumstances, the vector values are loaded into the matrix by column.

This can best be illustrated through some examples. These examples utilize array label sets called V2 with two items and V3 with three items, and vectors X2a, X2b, X2c, X3a and X3b defined below (based on the V2 and V3 array label sets):

X3a = [1, 2, 3]

X3b = [6, 7, 8]

X2a = [10, 20]

X2b = [12, 22]

X2c = [30, 40]

Expression	Resulting Array
Matrix(V2, V3 , X2a, X2b, X2c)	10 12 30 20 22 40
Matrix(V2, V3 , X3a, X3b)	1 2 3 6 7 8
Matrix(V2, V2 , X2b, X2c)	12 22 30 40

When using array constructors, you can also utilize two specialized variables: *row* and *col*. These variables can be used to define values based on the row and the column being referenced. Within constructors, these variables are defined as follows:

- If the array label set representing the row variable is based on a named set, *row* represents the the row number. Likewise, if the array label set representing the column variable is based on a named set, *col* represents the column number.
- If the array label set representing the row variable is an indexed set, *row* represents the actual item label. Likewise, if the array label set representing the column variable is an indexed set, *col* also represents the actual item label.

They are best illustrated through some examples. These examples utilize a named array label set called Set1 with 3 items (A, B and C), an indexed array

label set called Set2 with 4 items (1,2,3 and 4), and an indexed array label set called Set3 with 3 items (3,4,5).

Expression	Resulting Array
Vector(Set1,row)	1 2 3
Vector(Set3,row)	3 4 5
Vector(Set1, if(row>=2,1,0))	0 1 1
Vector(Set3, if(row>=2,1,0))	1 1 1
Matrix(Set1, Set2, row)	1 1 1 1 2 2 2 2 3 3 3 3
Matrix(Set1, Set2, col)	1 2 3 4 1 2 3 4 1 2 3 4
Matrix(Set2, Set2, if(col>=row,1,0)	1 1 1 1 0 1 1 1 0 0 1 1 0 0 0 1

One powerful use of array constructors is to take advantage of the ability of arrays to reference variables as arguments. For example,

Vector(days, if(row<=4, A[row], B[row]))

would create a vector of days in which the first 4 items would be drawn from the vector A, and the last 3 items would be drawn from the vector B.

Read more: Referencing an Item of an Array (page 736).

Warning: If you have elements with the names Row or Col, GoldSim will link to the element rather than use the keyword. Therefore, if you want to use these keywords, you should ensure that no elements exist in your model with these names.

Using a Vector as a Lookup Table

Two array functions are available in GoldSim whose primary application is to allow you to create dynamic lookup tables (in which the values for the independent and/or dependent variables are changing with time).

Although GoldSim provides a specialized Lookup Table element, this element does not support dynamic tables (all the values must be constant values).

Read more: Lookup Table Elements (page 263).

Using the array functions vIndex and vInterp provides a mechanism for creating a dynamic lookup table using a vector.

vIndex(vector, value) requires the first argument to be a vector, and the second argument to be a value (with the same dimensions as the vector). It searches for the specified value within the provided vector, and returns the dimensionless *index* at which the value is found. It treats the vector as a continuous function, linearly interpolating where necessary between the defined points in the vector, and as a result, the returned index value is not necessarily an integer. For example, if the vector's values were [3m, 4m, 5m, 6m], then vIndex(vector, 5.1m) would return an index value of 3.1. Similarly, vIndex(vector, 10ft) would return an index value of 1.048. If the array labels are indexed, the appropriate index value is returned; if the array labels are named, the first entry in the array has an index value of 1.

If the value is outside of the range of values in the vector, a fatal error is displayed. If the value exists multiple times in the vector (e.g., a sine function), the first matching value is used.

vInterp(vector, index) requires the first argument to be a vector, and the second argument to be a dimensionless index. It interpolates into the vector using the specified index value. It treats the vector as a continuous function, linearly interpolating where necessary between the defined values in the vector, and returns a value (with the same dimensions as the vector). For example, if the vector's values were [3m, 4m, 5m, 6m] then vInterp(vector, 3.1) would return a value of 5.1m.

If the index is outside of the range of indices for the vector, a fatal error is displayed.

These two functions can be used in conjunction to create a dynamic lookup table. To do so, you would create two vectors using the same array label set, which would typically be indexed (e.g., 1 through 10). One of the vectors (named VectorInd here) would contain the independent variables, and the other vector (named VectorDep here) would contain the corresponding dependent variables. The values of these could change dynamically. At any time, if you wanted to obtain the dependent variable for a particular value of the independent variable (say, X), you could do so using the following expression:

```
vInterp(VectorDep, vIndex(VectorInd, X))
```

This would look up the index of X in the vector representing the independent variable and return the corresponding value from the vector representing the dependent variable.

A reverse lookup into the same table could be carried out using this expression:

```
vInterp(VectorInd, vIndex(VectorDep, Y))
```

This would look up the location of Y in the vector representing the dependent variable and return the corresponding value from the vector representing the independent variable.

An example model which illustrates how these functions can be used to create dynamic lookup tables LookupTable_Dynamic.gsm) can be found in the General Examples/LookupTable folder in your GoldSim directory.

Manipulating Vectors and Matrices with Other Elements

Although you often will use Data elements to create arrays in your models, the real power of arrays is that they can be manipulated using other elements. In general, when you manipulate arrays in other elements, GoldSim carries out the calculations in parallel for each item of the array. GoldSim also provides a wide variety of special array functions which allow you to manipulate arrays

In this section, the various operators and functions that can be used to build expressions involving arrays are discussed. The manner in which various elements can manipulate array inputs is also described.

Manipulating Arrays in Expressions Using Mathematical Operators

The manner in which the standard operators in GoldSim can be used with arrays is summarized below:

Addition and Subtraction of Arrays. The addition (+) and subtraction (-) operators can be used between arrays if and only if the two arrays have the same dimensions and order, and are defined using the same set of array labels. GoldSim carries out the operation term-by-term and produces an array of the same order. For example, if A was a vector based on the set "Days", and B was a vector based on the set "Days", C=A + B would also produce a vector based on the set "Days". The first item of C would be the sum of the first item of A and the first item of B; the second item of C would be the sum of the second item of A and the second item of B, and so on. You cannot add or subtract an array to/from a scalar.

Multiplication or Division of an Array by a Scalar. The multiplication (*) and division (/) operators can be used between arrays and a scalar. Each item of the array is multiplied or divided by the scalar. For example, you could create the expression 2 * A, where A was a vector. The output of the Expression would be a vector, identical to the A vector (and its output attributes would need to be defined accordingly), except each item of the vector would be two times as great. Note that not only can you divide an array by a scalar, but you can also divide a scalar by an array. In both cases, the result is an array.

Multiplication and Division of Arrays of Same Order. When the multiplication (*) and division (/) operators are used between arrays that have the same order (and are defined using the same set of array labels), GoldSim carries out the operation term-by-term and produces an array of the same order as the original arrays. For example, if A was a vector based on the set "Days", and B was a vector based on the set "Days", C=A * B would also produce a vector based on the set "Days". The first item of C would be the product of the first item of A and the first item of B, etc.

Multiplying and/or Dividing the Rows or Columns of a 2-D Array (Matrix) by a 1-D Array (Vector). The following operations are supported in GoldSim for manipulating matrices: Matrix*Vector, Vector*Matrix, and Matrix/Vector. In these cases, the array label set of the Vector must match the array label set of either the rows or columns for the Matrix. If it matches that of the rows, then each row of the matrix is multiplied (or divided) by the corresponding term in the Vector. If it matches that of the columns, then each column of the matrix is multiplied (or divided) by the corresponding term in the Vector. If it matches both, the operation is carried out on the rows. In either case, the result is a Matrix of the same order as the original matrix.

Note: The operation described here is quite different from the linear algebra operation of multiplying a Vector by a Matrix (or vice versa), with the result being a Vector. That particular operation can be carried out using the specialized array function "Mult".

Read more: Array Functions (page 743).

Raising an Array to a Power. You can use the exponentiation operator (** or ^) to raise the items of an array to a power. GoldSim carries out the operation term-by-term and produces an array of the same order. For example, you could create an expression named C defined as A^2, where A was a vector. C must have the same order and set of array labels as A. The first item of C would be the first item of A squared, the second item of C would be the first item of A squared, etc.

Using Relational Operators with Arrays. Relational operators (e.g., >, <, =, ==, >=) can be used between arrays if and only if the two arrays have the same dimensions and order, and are defined using the same set of array labels. GoldSim carries out the operation term-by-term and produces an array of conditions. For example, if A was a vector based on the set "Days", and B was a vector based on the set "Days", the Expression C defined as A > B would produce a vector based on the set "Days". The first item of C would be the outcome (true or false) of the expression "first item of A" > "first item of B"; the second item of C would be the outcome (true or false) of the expression "second item of A" > "second item of B", and so on.

Relational operators can also be used between arrays and scalars. For example, if A was a vector based on the set "Days", and W was a scalar, the Expression D defined as A > W would produce a vector based on the set "Days". The first item of D would be the outcome (true or false) of the expression "first item of A" > "scalar value W"; the second item of D would be the outcome (true or false) of the expression "second item of A" >"scalar value W", and so on.

Using Arrays in If Statements. If statements can use mixtures of arrays and scalars. The rules for how such statements are interpreted are as follows:

1. The first argument (the condition) can be an array or scalar. If it is an array, then the second and third argument can either be arrays with the same set of array labels as the condition or scalars. Scalars are treated as arrays of identical values with the same set of array labels as the condition. If the first argument is an array, GoldSim does an item by item evaluation to construct the output array.
2. If the condition is scalar, and the second and third arguments are arrays, they must be of the same order, and the output has the same order as these arguments. Either one array or the other is output in its entirety as the result.
3. If the condition is a scalar, one of the latter two arguments can be a scalar, and one can be an array. The output of the If statement is then an array, and the scalar is treated as an array of identical values. Either one array or the other is output in its entirety as the result.

Read more: Entering and Editing Expressions in Input Fields (page 87).

Array Functions

Most of the built-in functions in GoldSim can also operate on arrays. In most cases, these functions carry out the operation term-by-term and produce an array of the same order as the input argument. For example, if A was a vector based on the set "Days", C=sin(A) would also produce a vector based on the set "Days". The first item of C would be the sine of the first item of A, the second item of C would be the sine of the second item of A, and so on.

Several functions (max, min, mod, tdist, tprob, bess, beta) accept multiple array arguments. For these functions, the arguments must all be scalars, or must be arrays of the same order that are defined using the same set of array labels.

GoldSim carries out the operation term-by-term and produces an array of the same order. For example, if A was a vector based on the set "Days", and B was a vector based on the set "Days", C=min(A, B) would also produce a vector based on the set "Days". The first item of C would be the minimum of the first item of A and the first item of B; the second item of C would be the minimum of the second item of A and the second item of B, and so on.

Two specialized functions (occurs and changed) cannot accept arrays as arguments.

If X and Y are arrays (of the same order and using the same set of array labels), the If(C, X, Y) function can manipulate the arrays in two different ways:

- If C is a scalar condition, the output of the If function is either the array X (if C is true) or the array Y (if C is false).
- If C is an array (of the same order and using the same set of array labels as X and Y), GoldSim will carry out a term-by-term if,then computation to produce the output array. That is, if the first item of C is true, the first item of the output would be equal to the first item of X, otherwise it would be equal to the first item of Y, and so on.

Read more: Built-in Functions (page 128).

In addition to the standard functions mentioned above, GoldSim also provides a wide variety of special array functions which allow you to manipulate arrays (accessible from the context menu for an input field).

These special array functions are listed below:

Function	Description	Result
GetItem(VC,n)	Returns the nth item in the vector.	scalar
GetItem(MC,n, m)	Returns the item in the nth row and mth column of the matrix.	scalar
GetRow(VC,n)	Returns the nth item in the vector. (If first argument is a vector, this function is identical to GetItem).	scalar
GetRow(MC,n)	Returns the nth row of the matrix.	vector
GetColumn(MC,m)	Returns the mth column of the matrix.	vector
GetRowCount(VC)	Returns the number of items in the vector.	scalar
GetRowCount(MC)	Returns the number of rows in the matrix.	scalar
GetColumnCount(MC)	Returns the number of columns in the matrix.	scalar
sumv(VC, X)	Sums the items of the vector. If the arguments are conditions, ORs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.	scalar

Function	Description	Result
prodv(VC, X)	Computes the product of the items of the vector. If the arguments are conditions, ANDs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.	scalar
minv(V)	Computes the smallest item in the vector.	scalar
maxv(V)	Computes the largest item in the vector.	scalar
meanv(V)	Computes the mean of the items in the vector.	scalar
sdv(V)	Computes the standard deviation of the items in the vector.	scalar
rowmin(V)	Computes the index of the smallest item in the vector (i.e., 1, 2, 3, etc.).	scalar
rowmax(V)	Computes the index of the largest item in the vector (i.e., 1, 2, 3, etc.).	scalar
sort123(V)	Sorts the items in the vector from smallest to largest.	vector
sort321(V)	Sorts the items in the vector from largest to smallest.	vector
dot(V1,V2)	Dot (inner) product of V1 and V2. The first vector is assumed to be a row vector and the second vector is assumed to be a column vector. The two vectors must be based on the same set of array labels.	scalar
vvmatrix(V1,V2)	Vector multiplication (to produce a matrix).The first vector is assumed to be a column vector and the second vector is assumed to be a row vector. This is equivalent to vector multiplication of V1 by the transpose of V2.	matrix[A,B], where A is the set of array labels for V1 and B is the set of array labels for V2
vIndex(V1,X)	Vector lookup. The second argument must be a value (with the same dimensions as the vector). It searches for the specified value within the provided vector, and returns a dimensionless *index* at which the value is found. It treats the vector as a continuous function, linearly interpolating where necessary between the defined points in the vector.	scalar

Function	Description	Result
vInterp(V1,n)	Vector interpolation. The second argument must be a dimensionless index. It interpolates into the vector using the specified index value. It treats the vector as a continuous function, linearly interpolating where necessary between the defined values in the vector, and returns a value (with the same dimensions as the vector).	scalar
sumr(MC, X)	Sums the items across each row of the matrix. If the arguments are conditions, ORs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.	vector[A], where A is the set of array labels for the rows in the original matrix
prodr(MC, X)	Computes the product of the items across each row of the matrix. If arguments are conditions, ANDs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.	vector[A], where A is the set of array labels for the rows in the original matrix
minr(M)	Computes the smallest item in each row of the matrix.	vector[A], where A is the set of array labels for the rows in the original matrix
maxr(M)	Computes the largest item in each row of the matrix.	vector[A], where A is the set of array labels for the rows in the original matrix
meanr(M)	Computes the mean of the items in each row of the matrix.	vector[A], where A is the set of array labels for the rows in the original matrix
sdr(M)	Computes the standard deviation of the items in each row of the matrix.	vector[A], where A is the set of array labels for the rows in the original matrix
sumc(MC, X)	Sums the items down each column of the matrix. If the arguments are conditions, ORs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.	vector[B], where B is the set of array labels for the columns in the original matrix

Function	Description	Result
prodc(MC, X)	Computes the product of the items down each column of the matrix. If the arguments are conditions, ANDs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.	vector[B], where B is the set of array labels for the columns in the original matrix
minc(M)	Computes the smallest item in each column of the matrix.	vector[B], where B is the set of array labels for the columns in the original matrix
maxc(M)	Computes the largest item in each column of the matrix.	vector[B], where B is the set of array labels for the columns in the original matrix
meanc(M)	Computes the mean of the items in each column of the matrix.	vector[B], where B is the set of array labels for the columns in the original matrix
sdc(M)	Computes the standard deviation of the items in each column of the matrix.	vector[B], where B is the set of array labels for the columns in the original matrix
trans(MC)	Transpose of matrix. trans(MC1[A,B]) = MC2[B,A]	matrix[B,A], where A is the set of array labels for the rows in the original matrix and B is the set of array labels for the columns
inv(M)	Inverse of matrix. inv(M1[A,B]) = M2[B,A]	matrix[B,A], where A is the set of array labels for the rows in the original matrix and B is the set of array labels for the columns
mult(M1,M2)	Matrix multiplication. The columns of the first matrix must be based on the same set of array labels as the rows of the second matrix. Mult(M1[A,B] , M2[B,C]) = M3[A,C].	matrix[A,C], where A is the set of array labels for the rows in matrix M1 and C is the set of array labels for the columns in matrix M2
mult(M,V)	Matrix times vector. The vector is automatically treated as a column vector. The vector must be based on the same set of array labels as the columns of the matrix. Mult(M[A,B] , V[B]) = V[A]	vector[A], where A is the set of array labels for the rows in matrix M

Function	Description	Result
mult(V,M)	Vector times matrix. The vector is automatically treated as a row vector. The vector must be based on the same set of array labels as the rows of the matrix. Mult(V[A] , M[A,B]) = V[B]	vector[B], where B is the set of array labels for the columns in matrix M

V, V1, V2: Vector of values
VC: Vector. Can be value or condition.
M, M1, M2: Matrix of values
MC, MC1, MC2: Matrix. Can be value or condition.
X: Scalar value.
n,m: a dimensionless value; can be specified as a number, equation or a link. Real values are rounded to the nearest integer.

Note that the GetItem, GetRow, and GetColumn functions can, in most cases. more easily be implemented by referencing items of an array directly using variables. That is, if A was based on a named set or an indexed set that started with 1, GetItem(A, X) is identical to A[X].

Note, however, that in his example, if A was an indexed set that did not start with 1, GetItem(A, X) would not be identical to A[X]. GetItem(A,X) would return the Xth row. A[X] would return the item whose label was X. These are only the same if the indexed set starts with 1.

Another difference is that the first argument of the GetItem, GetRow and GetColumn functions can itself be an expression. To implement this using arrays with variable arguments would therefore require you to create a second element. That is, to reproduce this expression:

GetItem(A * B, X),

you would first need to create a vector C = A*B, and then reference C[X].

Read more: Referencing an Item of an Array (page 736).

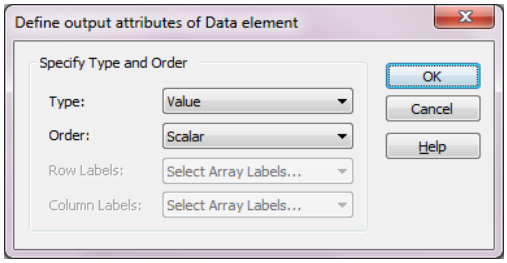
Elements That Can Manipulate Arrays

Many of the GoldSim elements can manipulate and produce vectors and matrices. For example, you can sum a number of matrices using the Sum element, or compute the peak value of each item of a vector using the Extrema element.

In general, if the outputs for an element are an array, its inputs must be arrays (defined by the same sets of array labels), and GoldSim carries out the element's calculations in parallel for each item of the array.

Note: If an input field of an element requires a vector, and the vector only has a single item, GoldSim allows you to enter a scalar as the input. Of course, if you then add items to the set of array labels (such that it consists of more than one item), the scalar input will no longer be valid.

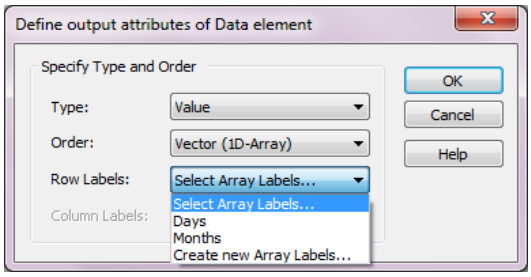
In order for an element to output an array, you must specify that it is to be a vector or a matrix when you define its output attributes (by pressing the **Type…** button in the element's properties dialog). The dialog for specifying output attributes for an element looks like this:



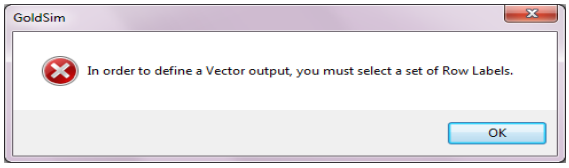
The **Order** drop-list provides three choices: "Scalar" (the default), "Vector (1D-Array)", and Matrix (2D-Array)":

If you wish to create a vector, select "Vector (1D-Array)". If you wish to create a matrix, select "Matrix (2D-Array)". If you select "Vector (1D-Array)", the **Row Labels** field becomes available. If you select "Matrix (2D-Array)", both the **Row Labels** and **Column Labels** fields become available.

These two fields contain lists of all of the sets of array labels defined in your model:



If you have defined the **Order** as a vector or a matrix (i.e., if the **Row Labels** or **Column Labels** fields are active), you must select a set of array labels (or GoldSim will display a warning message when you try to close the Output Attributes dialog):

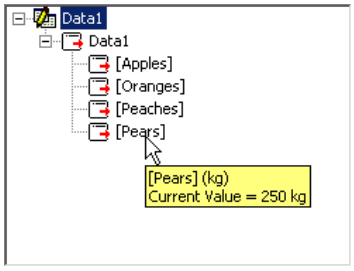


Within the drop-list for the Row and Column labels is the choice to "Create new Array Labels". If you click this option, the dialog for defining new array labels will be displayed. After leaving the Array Labels dialog, you will be returned to the dialog for defining output attributes.

Read more: Creating and Editing Array Labels (page 728).

Viewing Results for Arrays

If an element has an output which is an array, you can view the values of specific items of the array by holding your cursor over the item in a browser or the output interface.



Note: Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

You can also view array output results in table form or chart form (as bar charts).

Read more: Viewing Array Results (page 648).

Copying Array Elements Between Models

When you copy an element defined using a set of array labels (i.e., an element with vector or matrix outputs) to another model, you must ensure that the set(s) of array labels referenced by the element in the two models are consistent. In particular, GoldSim imposes the following rules:

- If the set(s) of array labels referenced by the element already exist in the model into which it is being pasted, the number of items in the set(s) must be identical in both models. Note that it is not necessary that the set labels be identical, as long as the number of items is identical.
- If the set(s) of array labels referenced by the element do not exist in the model into which it is being pasted, GoldSim will automatically create them in the new model.

Read more: Copying Elements Between Model Files (page 108).

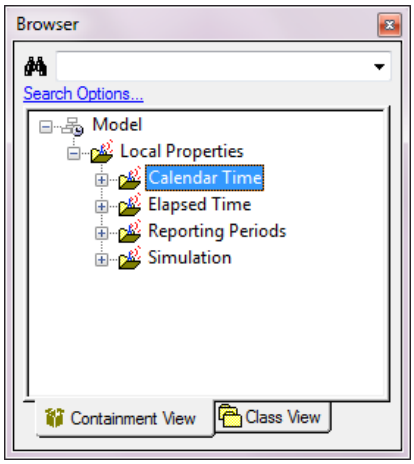
Understanding Locally Available Properties

Locally available properties are special attributes of some elements in GoldSim. Locally available properties are similar to element outputs in that they have a data type (e.g., value, condition), order (e.g., scalar, vector) and dimensions (i.e., units). However, they do not appear as outputs of the element to which they belong. Rather, they are only visible in browsers (the main browser, or the Insert Link browser).

Note: Locally available properties are only shown in the main browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

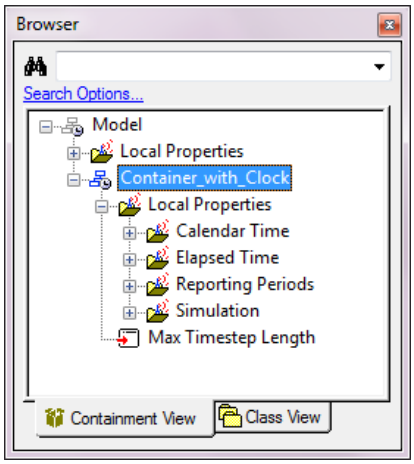
Locally available properties derive their name from the fact that they may only be available, or they may take on different values (i.e., be over-ridden), in "locally available" parts of your model (e.g., within particular Containers).

Containers are often providers of locally available properties. In fact, the Run Properties are actually locally available properties belonging to the Model Container:



Read more: Understanding and Referencing Run Properties (page 445).

These properties are "broadcast" to the entire model. A Run Property (like ETime) is an example of a type of locally available property that is available throughout a model, but can take on different values in different parts of the model. In particular, some of the Run Properties can be over-ridden locally if a Container is defined to have an internal clock (a local timestep). In such a case, the Container with the internal clock also possesses Run Properties:



If you define an internal clock for a Container, and then within that Container, reference ETime, GoldSim will automatically connect to the "locally available" ETime (the ETime in the Container), rather than the global ETime (the ETime associated with the "root" or Model Container). Hence, the actual locally available property that it linked to is a function of where it is referenced from.

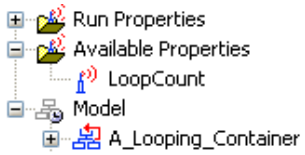
Read more: Specifying Containers with Internal Clocks (page 434).

Some locally available properties are only available in specific locations. A good example of this is the loop count for a Looping Container.

Read more: Using Looping Containers (page 904).

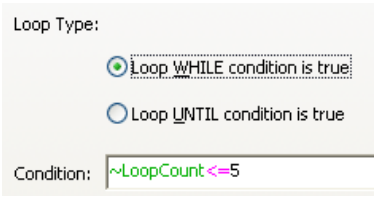
Within the looping Container itself (or in its property dialog), it is often necessary to reference the loop count. However, this variable has no meaning whatsoever outside of the looping Container. As a result, it can only be referenced inside the Container (including in its property dialog). If you right-click in an input field within the looping Container and select **Insert Link** to

view the Insert Link browser, the loop count will appear in an Available Properties folder:



If you invoke the Insert Link browser from outside of the Container, however, the loop count is not visible and cannot be referenced.

Locally available properties are referenced in expression by adding the ~ prefix. For example, when you insert the loop count for a looping Container into a field, it appears as follows:



Note: Run Properties represent a special instance of locally available properties that do not require the "~" prefix. These properties can be referenced directly (e.g., as ETime, or DayofWeek).

Modeling Aging Chains

In some situations, it is necessary to keep track of the age structure of a stock of material or objects. For example, you may want to track the number of people in each of a number of age groups, the number of people in a company at different experience levels (e.g., new hire, experienced, expert), or the number of trucks of different age groups on the road.

To model such a situation, you cannot use a single Stock (e.g., a Reservoir). Rather, you must disaggregate the total stock into multiple categories (referred to as cohorts). Each cohort "graduates" to the next cohort over time (and can only move in one direction). However, each cohort may grow and shrink for other reasons (e.g., if you were modeling a population of people, a particular category could grow due to immigration, and shrink due to emigration and death). Note that these rates of growing and shrinking are likely a function of the specific cohort (e.g., death rates).

Depending on your conceptual model, there are three fundamental ways to model such a chain:

- Using a series of Reservoirs.
- Using a series of Material Delays.
- Using a series of Integrators.

These three methods are discussed briefly in the following sections.

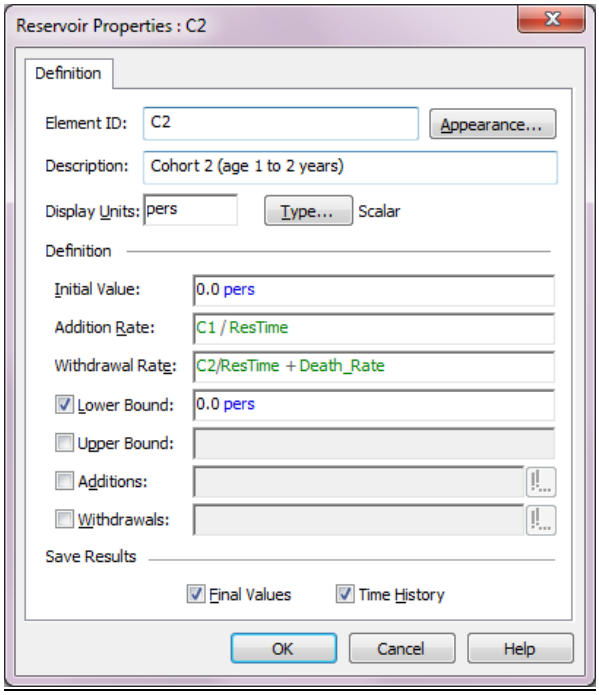
An example file which illustrates how aging chains can be modeled (Aging_Chain.gsm) can be found in the General Examples folder in your GoldSim directory.

Modeling Aging Chains Using a Series of Reservoirs

Perhaps the simplest way of modeling aging chains is to create a series of Reservoirs. The transition rate from one cohort (say C2) to the next, is set equal to C2/ResTime (a first-order rate), where C2 is the quantity in cohort 2, and ResTime is the average residence time in cohort 2.

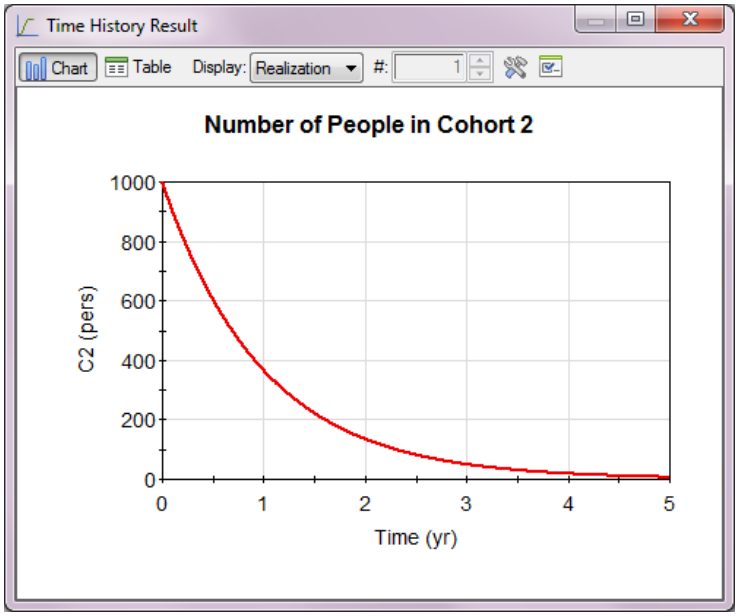
Read more: Reservoir Elements (page 236).

In such a situation, the Reservoir for cohort 2 would look like this:



In this example, the cohort represents people aged 1 to 2 years. Hence, the ResTime is equal to 1 year. The rate of additon is C1/ResTime, and the rate of withdrawal is C2/ResTime plus the loss rate due to death.

It is important to understand that due to the nature of a Reservoir, such a representation results in a behavior which is typically inappropriate for modeling specific age groups. To illustrate this, let's assume 1) that cohort 2 starts with 1000 one-year olds; 2) there is no addition to the cohort (C1 = 0)), and 3) there is no death rate. Cohort 2 only decreases due to graduations to the next cohort. Under these circumstances, the number of people in cohort 2 would look like this:



As can be seen, rather than staying at 1000 for the first year, and then abruptly changing to 0 (when all of the one-year olds actually become 2 and hence should immediately graduate to the next cohort), the number of one-year olds gradually decays over a period of 5 years. As a result, modeling an aging chain in this manner is most appropriate when the residence time in the cohort represents an average time, with some people leaving earlier and some later. This is the case, for example, if the cohort represents a group of employees (e.g., inexperienced), rather than a specific age group (e.g., between 1 and 2 years old). One of the other two methods of modeling aging chains (using Material Delay or Integrators with discrete pushes) should generally be used when modeling specific age groups.

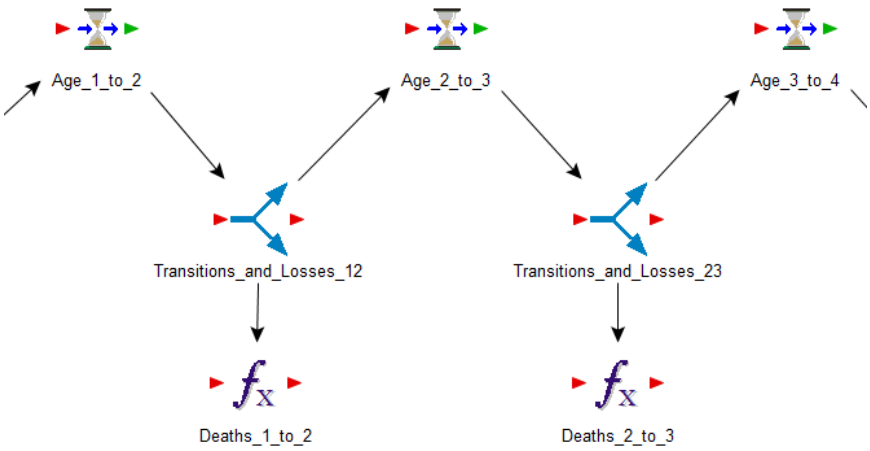
An example file which illustrates how aging chains can be modeled (Aging_Chain.gsm) using various methods can be found in the General Examples folder in your GoldSim directory.

Modeling Aging Chains Using a Series of Material Delays

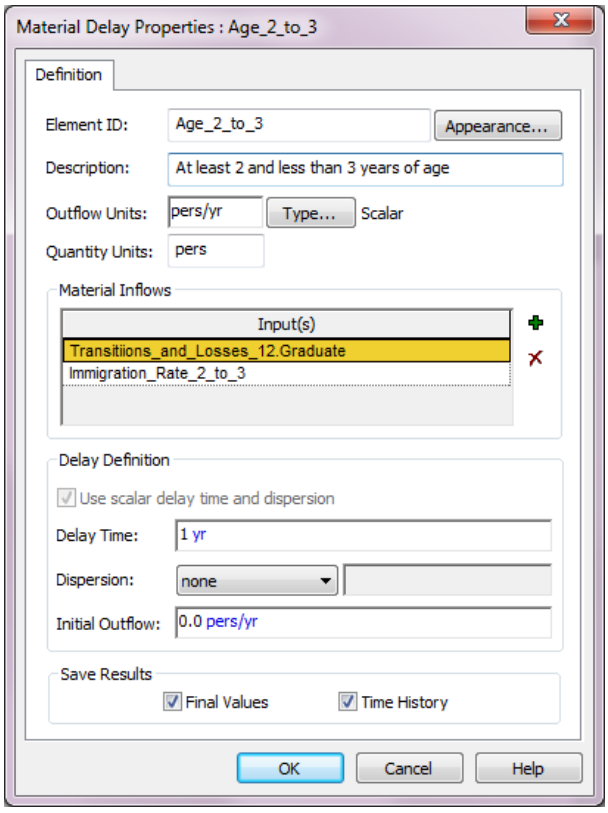
When modeling an aging chain consisting of specific age groups (e.g., 0 – 1 years, 1 – 2 years, etc.), it is often most appropriate to model the chain using a series of Material Delays.

Read more: Material Delay Elements (page 300).

Each Delay has a fixed delay time (with no dispersion) corresponding to the age cohort (e.g., 1 year). As such, the residence time in the cohort does not represent an average; it is an exact time required to "transit" the cohort. The structure of the model would typically look something like this:



Note that each Delay does not flow into the next Delay in the series. Rather, it flows through a Splitter, so that losses (e.g., due to deaths and emigration) can be accounted for prior to graduating to the next cohort. Of course, there could also be additional flows into a cohort (due to immigration). The Material Delay dialog for a typical cohort for this simple example (including an immigration inflow) would look like this:



An example file which illustrates how aging chains can be modeled (Aging_Chain.gsm) using various methods can be found in the General Examples folder in your GoldSim directory.

Modeling Aging Chains Using Integrators with Discrete Pushes

In some types of aging chains, you would like to intantaneously "graduate" all the members of one cohort to the next cohort at a specific time (e.g., every year), and repeat the process for all cohorts. For example, if you had 6 age groups, at the beginning of each year, you might want to transition all the age groups to the next cohort (e.g., cohort 5 graduates to cohort 6, cohort 4 graduates to cohort 5, cohort 3 graduates to cohort 4, etc.).

To facilitate this, GoldSim provides specialized capabilities for Discrete Change and Integrator elements.

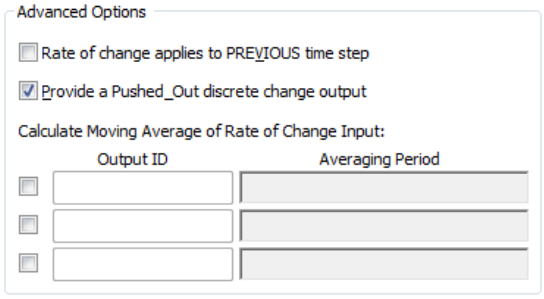
Read more: Integrator Elements (page 228); Discrete Change Elements (page 352).

One of the Instruction options for a Discrete Change transaction is a "Push":

Push instructions can subsequently be processed by Integrators (by specifying the discrete change output as an input in the **Discrete Change** input field of the Integrator).

Note: Push discrete change outputs can only be processed by Integrator elements. Splitters and Allocators can also accept Push discrete changes, as they don't actually process them; they simply pass them on (preserving the instruction). However, directing a Push discrete change to any other element (e.g., a Reservoir) will result in a fatal error.

When processing a Push discrete change signal, you must specify (under the Advanced options for an Integrator) that the Integrator itself, in turn, provides a"Pushed_Out" discrete change output:



Doing so adds a new output called "Pushed_Out" to the element.

When an Integrator receives a Push discrete change signal (and the "Provide a Pushed_Out discrete change output" is checked), the following occurs:

1. The current value of the Integrator is instantaneously "pushed out" and replaced with value specified by the incoming discrete change signal.
2. The element produces a discrete change output named "Pushed_Out"that has a "Push" Instruction and a Value equal to the value of the Integrator prior to receiving the discrete change signal. This can subsequently be directed to another Integrator.

By creating a series of Integrators connected by Push discrete changes (triggered, for example, once per year), you can create an aging chain that intantaneously "graduates" all the members of one cohort to the next cohort at a specific time (e.g., every year).

Note: When a scalar Integrator receives a Push discrete change signal and the "Provide a Pushed_Out discrete change output" is NOT checked, the discrete change is treated as if it had an Add instruction (since it has nowhere to "push" the current value). Hence, the Integrator simply accumulates the changes.

An example file which illustrates how aging chains can be modeled (Aging_Chain.gsm) using a series of scalar Integrators can be found in the General Examples folder in your GoldSim directory.

Although this type of approach can represent specific age groups very accurately, it can be cumbersome if there are a large number of groups. An alternative to this approach that is much better at simualting a large number of age groups is to use an Integrator *array* that processes discrete pushes.

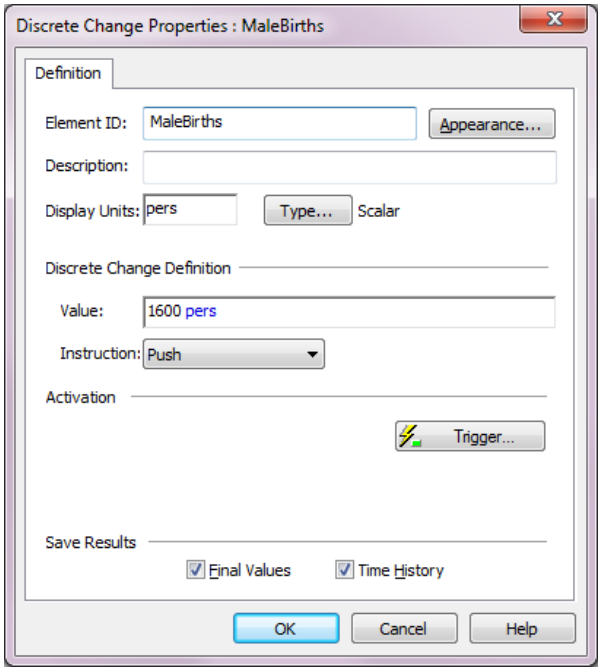
In this approach, instead of using a series of Integrators, you use a single Integrator that is a vector. A discrete change signal with a Push instruction is sent to the Integrator. When it receives the instruction, the Value associated with the discrete change is "pushed" into the first item of the vector. Each value in the vector is then pushed downward to the next item of the vector. That is, the value that was in item 1 is pushed to item 2; the value that was in item 2 is pushed to item 3, and so on.

If "Provide a Pushed_Out discrete change output" is checked for the Integrator, the element produces a discrete change output named "Pushed_Out" that has a "Push" Instruction and a Value equal to the value of the last item in the Integrator vector prior to receiving the discrete change signal. This can subsequently be directed to another Integrator. If "Provide a Pushed_Out discrete change output" is NOT checked for the Integrator, the final item in the Integrator vector accumulates all of the "pushes" that it receives.

When implementing such a vector aging chain, the Discrete Change element that generates the initial "push" into the Integrator must be defined in a specific manner. In particular, it must be defined as follows:

- The Instruction must be "Push";
- The Value must be a scalar value; and
- The Type must be defined to match the Type of the Integrator that will receive the discrete change. Hence, although the Value itself is a scalar (since it represents the value being pushed into the first item of the vector), the Discrete Change element must be defined as a vector (as it will be an input to a vector Integrator).

An example of this is shown below:



This approach allows you to replace a series of Integrators with a single vector Integrator, and therefore facilitates simulation of chains with a large number of cohorts

An example file which illustrates how aging chains can be modeled (Aging_Chain.gsm) using a vector Integrator can be found in the General Examples folder in your GoldSim directory.

This approach can be extended to utilize a matrix Integrator. In this case, a discrete change signal (whose Value is defined as a vector) with a Push instruction is sent to the Integrator. When it receives the instruction, the Value associated with the Discrete Change is "pushed" into the first *row* of the marix. Each row in the matrix is then pushed downward to the next row of the matrix. That is, the values that were in row 1 are pushed to row 2; the values that were in row 2 are pushed to row 3, and so on.

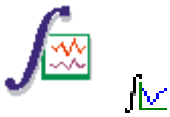
If "Provide a Pushed_Out discrete change output" is checked for the Integrator, the element produces a discrete change output named "Pushed_Out"that has a Push Instruction and a Value equal to the value of the last row in the Integrator matrix prior to receiving the Discrete Change signal. This can subsequently be directed to another Integrator. If "Provide a Pushed_Out discrete change output" is NOT checked for the Integrator, the final row in the Integrator matrix accumulates all of the "pushes" that it receives.

When implementing such a matrix aging chain, the Discrete Change element that generates the initial "push" into the Integrator must be defined in a specific manner. In particular, it must be defined as follows:

- The Instruction must be "Push";
- The Value must be a vector having the same Array Label Set as the columns of the matrix into which it is being pushed ; and
- The Type must be defined to match the Type of the Integrator that will receive the Discrete Change. Hence, although the Value itself is a vector (since it represents the values being pushed into the first row of

the matrix), the Discrete Change element must be defined as a matrix (as it will be an input to a matrix Integrator).

Solving Convolution Integrals



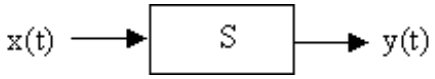
Convolution elements are highly specialized elements in GoldSim that solve *convolution integrals*. Convolution integrals have many important applications in engineering, science and business.

The Convolution element is discussed in detail below.

What is a Convolution Integral?

Before describing how to use a Convolution element, it is first important to understand exactly what the element does mathematically, and what this mathematical operation represents conceptually.

Consider a dynamic system, in which an input signal, say x(t), enters a "black box", S, and is output as y(t):



Mathematically, we represent the "black box" system as S:

$$y(t) = S[x(0 \text{ to } t)]$$

where y(t) represents the output signal at time t, and x(0 to t) represents the time history of the input signal. If the "black box" represents a convolution, it means that S takes on the following mathematical form:

$$y(t) = S[x(0 \text{ to } t)] = \int_0^t x(\tau)\, h_\tau(t-\tau)\, \delta\tau$$

In this equation $h_\tau(t - \tau)$ is the *transfer function* (also referred to as the *impulse response function*). $t - \tau$ is referred to as the *lag*, as it represents the time lag between the input and the output time. The transfer function is given a subscript τ to indicate that the function itself may take on different forms at different points in time. That is, the response to an input impulse at $t = \tau_1$ could have a different form than the response to an impulse at $t = \tau_2$.

Convolution elements require that you supply the input signal, x(t), and the transfer function, $h(t - \tau)$. The element then computes the output signal, y(t). Note that the convolution integral is a linear operation. That is, for any two functions $x_1(t)$ and $x_2(t)$, and any constant a, the following holds:

$$S[x_1(t) + x_2(t)] = S[x_1(t)] + S[x_2(t)]$$

$$S[ax_1(t)] = a\, S[x_1(t)]$$

Having defined mathematically what a convolution integral does, let us now try to understand what it represents conceptually. Perhaps the simplest way to think about the convolution integral is that it is simply a linear superposition of response functions, $h(t - \tau_i)$, each of which is multiplied by the impulse $x(\tau_i)\delta\tau$.

A more common way to interpret the convolution integral is that the output represents a weighted sum of the present and past input values. We can see this if we write the integral in terms of a sum (and assume here that the system is discretized by a single unit of time):

$$y(t) = x(0)h(t) + x(1)h(t-1) + x(2)h(t-2) + \ldots$$

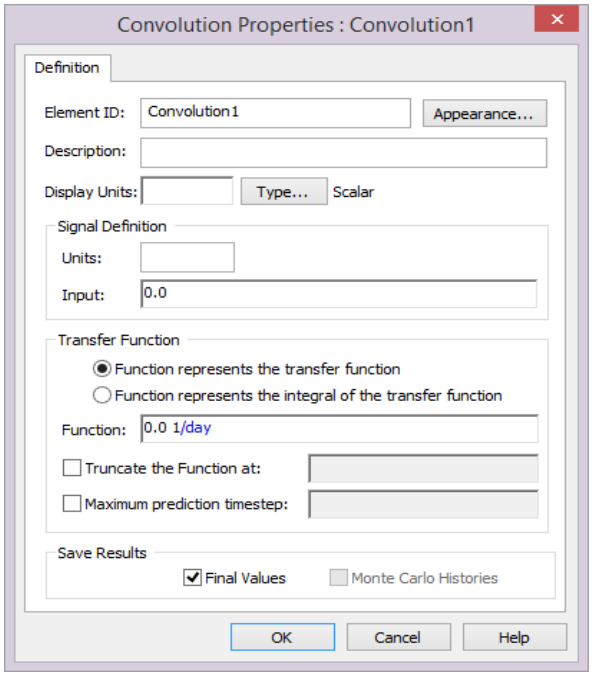
The convolution operation may also be thought of as a *filtering* operation on the signal x(t), where the transfer function is acting as the filter. The shape of the

transfer function determines which properties of the original signal x(t) are "filtered out."

Note that if the transfer function was a constant, the integral would collapse and the output signal would just be a scaled version of the input signal. Integration is necessary when the transfer function is not constant. Conceptually, this indicates that the system has memory and does not respond instantaneously to a signal. Rather, the response is delayed and spread out over time.

Using the Convolution Element

The property dialog for a Convolution element looks like this:



Convolution elements have a single output. You can specify these attributes by pressing the **Type…** button. By default, a new Convolution element is a scalar, dimensionless value. You can also use Convolution elements to operate on and/or create vectors and matrices.

Read more: Using Vectors and Matrices (page 726); Convolution Elements Defined as Arrays (page 762).

The **Display Units** field refers to the display units of the output signal (which may have different dimensions than the input signal).

You can save the results for a Convolution element by clicking **Final Values** and/or **Time Histories**.

Under *Signal Definition*, you first specify the **Input** signal. You must also specify the dimensions of the input signal (by defining **Units**). This facilitates error checking. The **Input** signal will be a direct or indirect function of time.

You then define the Transfer **Function**. In some cases, it may be easier to define the integral of the transfer function, rather than the transfer function itself, and the two radio buttons on the dialog allow you to choose between these two options.

The transfer function essentially "spreads out" or "disperses" a unit impulse over time. In many cases, the transfer function may be a continuous function with a long "tail". That is, 99% of output produced by an impulse may be generated within 10 days of the impulse, with the remainder being spread out

over the remaining 1000 days. Such a transfer function can have a significant computational impact on your model. Hence, GoldSim provides a way to "cut off" a transfer function after a certain lag time. If you check the **Truncate the Function at** checkbox, you can then enter a time after which the transfer function will be truncated. In the above example, you might, for example, specify this as 10 days. This input can be a function of time (e.g., if the transfer function changes with time, you may need to adjust the truncation time accordingly).

The Convolution element works by generating and continuously adding to a projected future history of its output. At each timestep the incremental input is convolved with the transfer function and added to that output history. The history stores projected results for every scheduled timestep of the model, and also inserts additional points where necessary to ensure that the difference between successive time points is never greater than 10% of the **Truncate the Function at** value.

However, if your model generates unscheduled updates and if the duration of the transfer function is short, it is possible that the projected future history will not have enough time points to deliver accurate results. In this situation you can use the optional input **Maximum prediction timestep** to ensure sufficient accuracy of this history. The must be specified as a time, and cannot have any links. It should be short enough so that it can reasonably approximate the shape of the transfer function.

Note that in the definition of a convolution integral, the transfer function is mathematically defined not as a function of time, but as a function of a time difference, $(t - \tau)$, or lag time. In order to write such an equation in GoldSim, you reference the time difference $t - \tau$ using a special locally available property of the Convolution element called "Lag". It must be referenced as "~Lag".

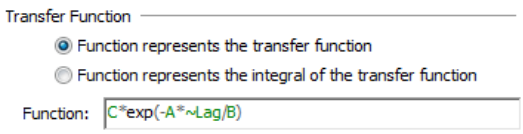
Note: As indicated by the way it is referenced, the Lag is an example of a locally available property. As such, it only has meaning inside the Transfer **Function** field, and cannot be referenced anywhere else.

Read more: Understanding Locally Available Properties (page 750).

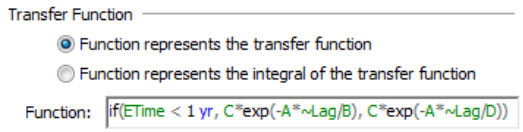
For example, if your transfer function was the following equation:

$$h(t-\tau) = Ce^{-a(t-\tau)/b}$$

then you would reference the lag time $(t - \tau)$ as "~Lag":



The transfer function does not have to be time-invariant. That is, the function can be defined to change as a function of time. Referring to the example above, if the response was different at later times in the simulation (e.g., after 1 yr), the transfer function could be written as follows:



GoldSim checks to ensure that the dimensions of the various inputs and outputs are related as follows:

Output = Transfer Function * Input Signal * time

If the integral of the transfer function is specified, GoldSim checks to ensure that the dimensions of the various inputs and outputs are related as follows:

Output = Integral of Transfer Function * Input Signal

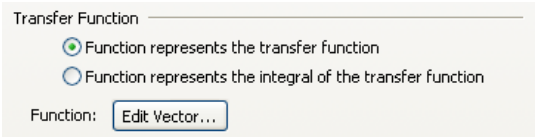
For example, if the element display units were specified as $ and the input signal display units were specified as g/yr, then GoldSim would require that the transfer function had dimensions of currency/mass (e.g., $/g).

Several points should be noted regarding the numerical implementation of the Convolution element within GoldSim:

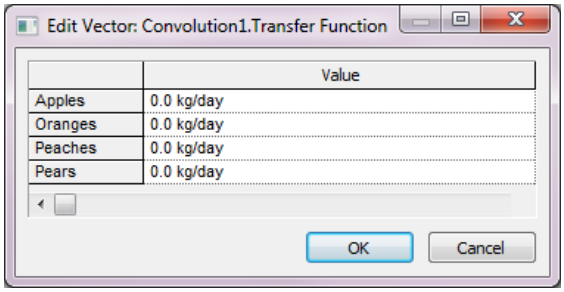
- During the numerical integration, GoldSim does not actually start the integration from 0. Instead, it starts the integration from a very small number. This allows the transfer function to be a function of the inverse of the lag without causing numerical problems (i.e., divide by zero errors)
- The numerical integration approach used by GoldSim assumes that the input function varies linearly between time steps; and
- To have acceptable accuracy, the simulation timestep should be short relative to the rate of change of the transfer function. Note, however, that your model may run much slower if the timestep is too small, as the computational effort for solving a Convolution element goes up as the square of the number of timesteps.

Convolution Elements Defined as Arrays

A convolution element can be defined to output an array. In this case, the dialog appears like this:



In this case, the Input Signal must be specified as an array. Pressing the **Edit Vector…** button (or **Edit Matrix…**button if it is a matrix) provides access to a dialog for defining the scalar transfer function for each separate item of the array:



Examples of the Use of the Convolution Element

In order to fully understand how to use Convolution elements, it is helpful to explore several examples. The three examples discussed here can be found in the General Examples folder of your GoldSim directory, in a file named Convolution.gsm.

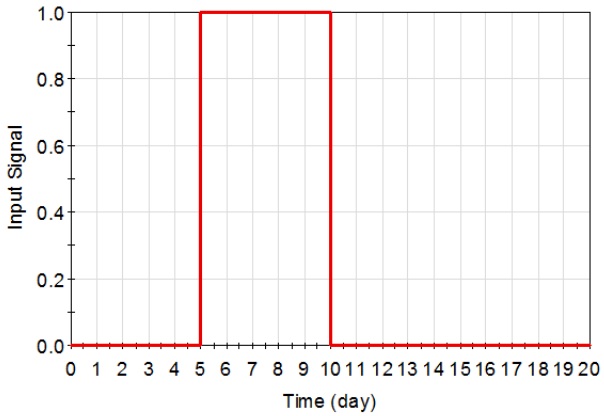
We examine three simple examples below:

- A simple filter;

- Computing costs associated with warranties; and
- Computing response of a structure due to an earthquake.

Convolution Example: A Simple Filter

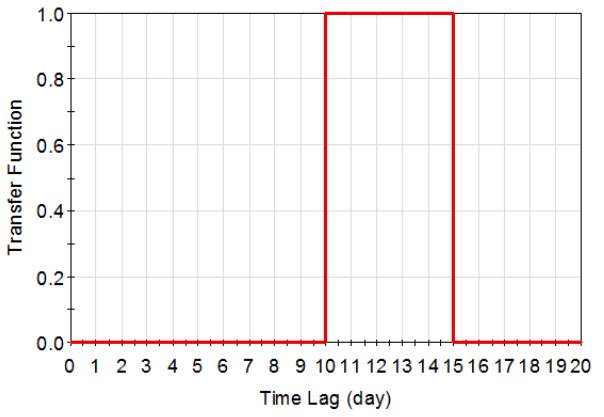
This example shows how a simple input signal can be filtered to alter its shape. In this example, the input signal is a square "boxcar" function:



Mathematically, this can be expressed in GoldSim as follows:

If(ETime < 5 days OR ETime > 10 days, 0, 1)

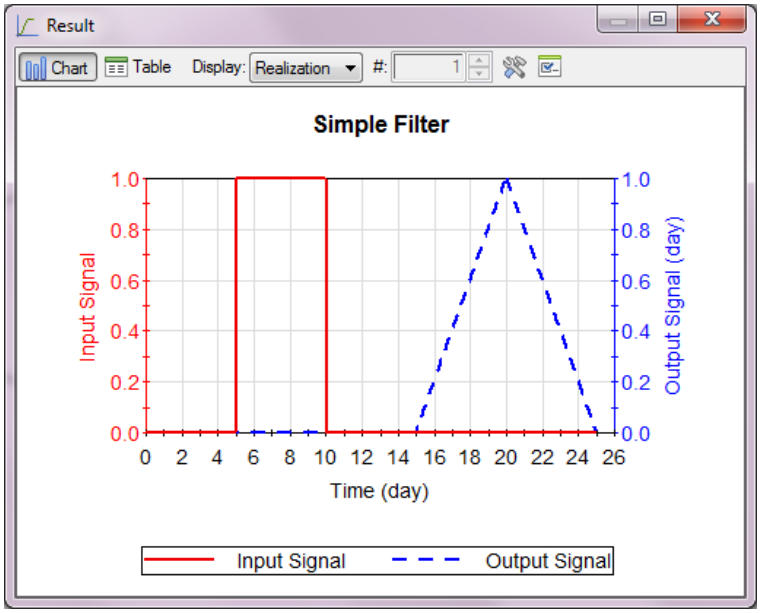
The Transfer function is also a square "boxcar" function, in which any impulse is delayed for 10 days, and then spread out evenly over 5 days:



Mathematically, this can be expressed in GoldSim as follows:

If(~Lag < 10 days OR ~Lag > 15 days, 0, 0.2)

This input signal and transfer function produce the following output signal:

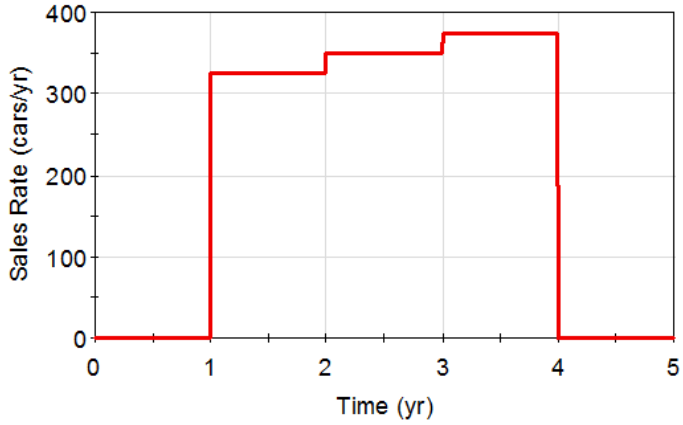


Note that the output signal in this case has dimensions of time (since the input signal and the transfer function were defined as being dimensionless).

Convolution Example: Warranty Costs

This example shows how warranty costs for a product (e.g., a vehicle) can be computed based on a product sales rate and the expected warranty cost time history for a product using a Convolution element.

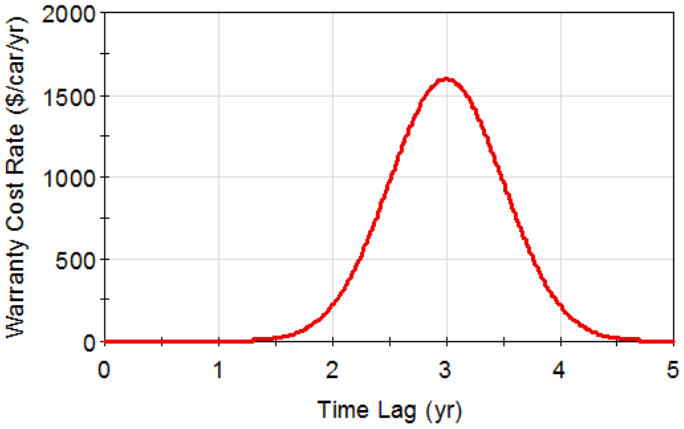
In this example, the input signal is the sales rate of the vehicle:



Sales will start one year in the future, and ramp slowly up for three years. The vehicle is only assumed to be produced for three years.

This can be entered into GoldSim as a Time Series.

The Transfer function represents, on average, how warranty costs are expected to be incurred in the future (per vehicle):



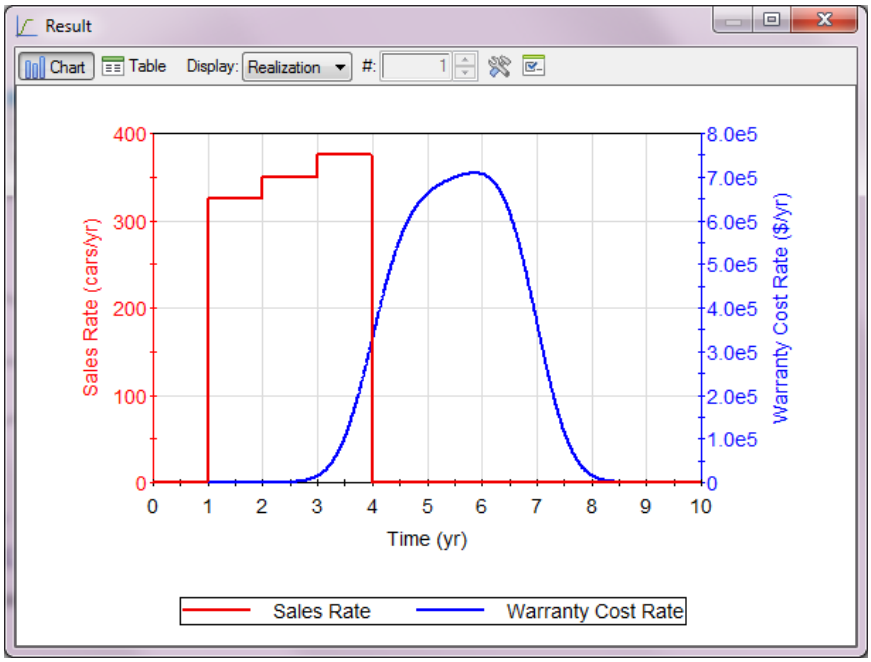
This curve indicates that warranty costs start to be incurred about 1.5 years after the sale, reach a peak at 3 years after the sale, and no more costs can be expected to occur after about 4.5 years after the sale. Note that the units on the transfer function are $/car/yr (the curve integrates to 2000 $/car).

The output signal will then have dimensions of $/yr:

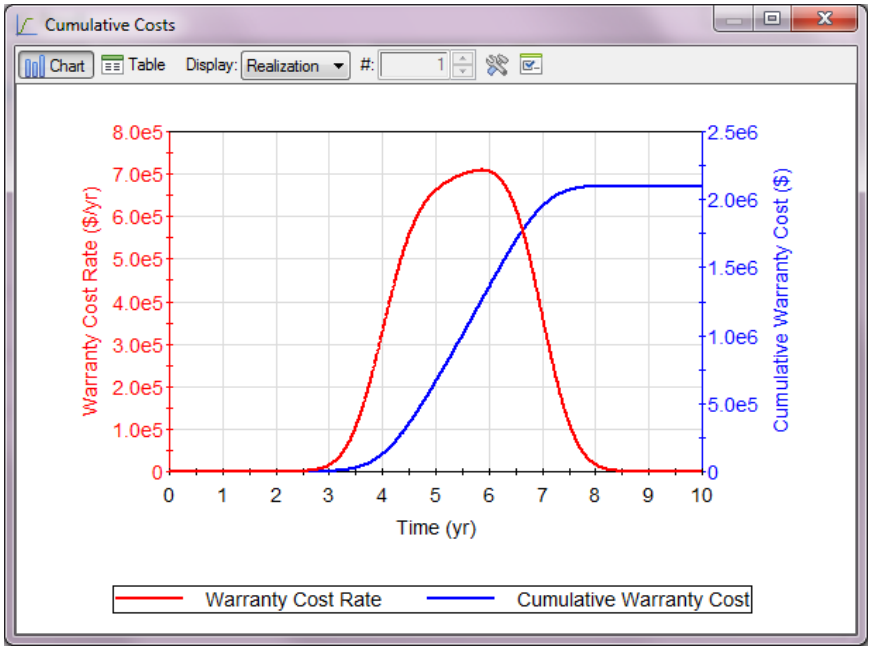
Output = Transfer Function * Input Signal * time

Output = $/car/yr * cars/yr * yr = $/yr

That is, the output represents the rate at which we can expect to incur warranty costs for all of the vehicles sold:



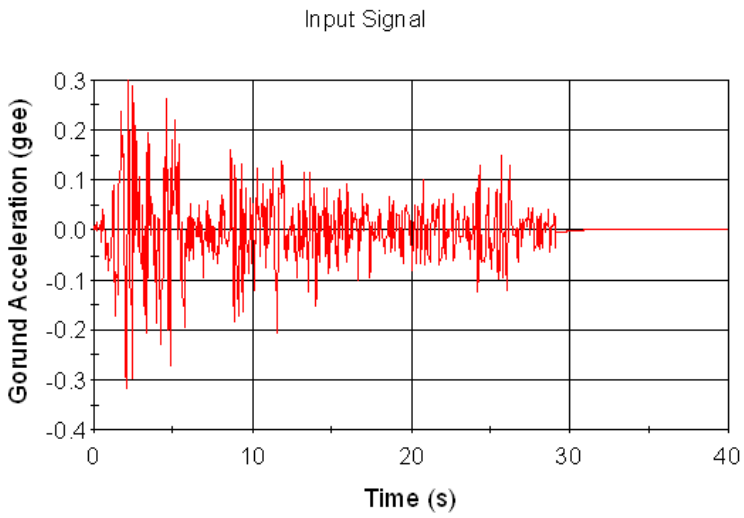
If we integrate the warranty cost rate (using an Integrator), we can compute the cumulative warranty costs:



Convolution Example: Earthquake Response

This example shows how a Convolution element can be used to simulate how a structure responds to an earthquake. The input signal is an accelerogram (a time history of ground accelerations). The output signal is the displacement of a building (illustrating how the building oscillates from side to side in response to the earthquake).

The input signal (the accelerogram) is taken from an actual earthquake in Southern California in 1940 (the El Centro earthquake):

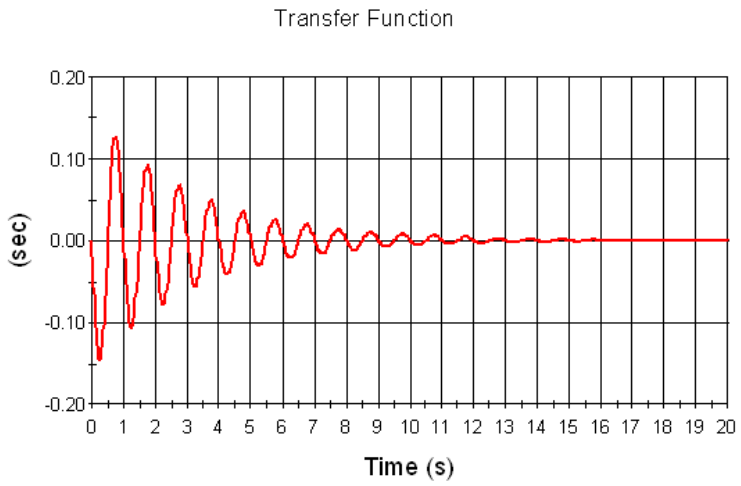


As can be seen, the earthquake lasted about 30 seconds.

For simplicity, the building is treated as a damped spring. The transfer function under such an assumption is represented by the following equation:

$$\frac{-e^{[-n(t-\tau)]}}{W_d}\sin[W_d(t-\tau)]$$

where n is the coefficient of decay (in radians/sec), and W_d is the damped frequency (in radians/sec). A plot of this transfer function looks like this:



The transfer function represents the ratio of a displacement response to a "pulse" of ground velocity, and has dimensions of time.

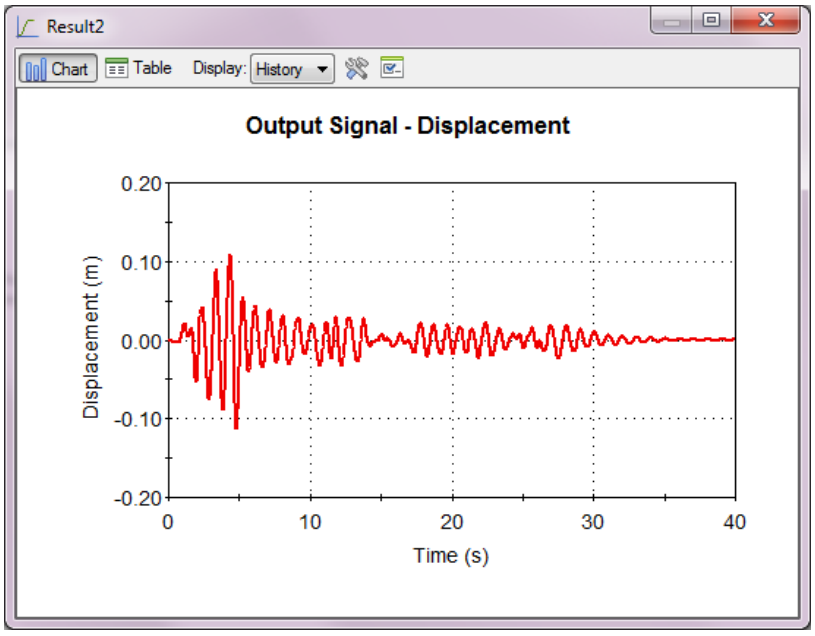
This curve indicates that once excited by an impulse, the building oscillates with the oscillations dying out within about 15 seconds.

The output signal will have dimensions of length:

Output = Transfer Function * Input Signal * time

Output = sec * m/sec^2 * sec = m

That is, the output represents the displacement of the building in response to the earthquake:



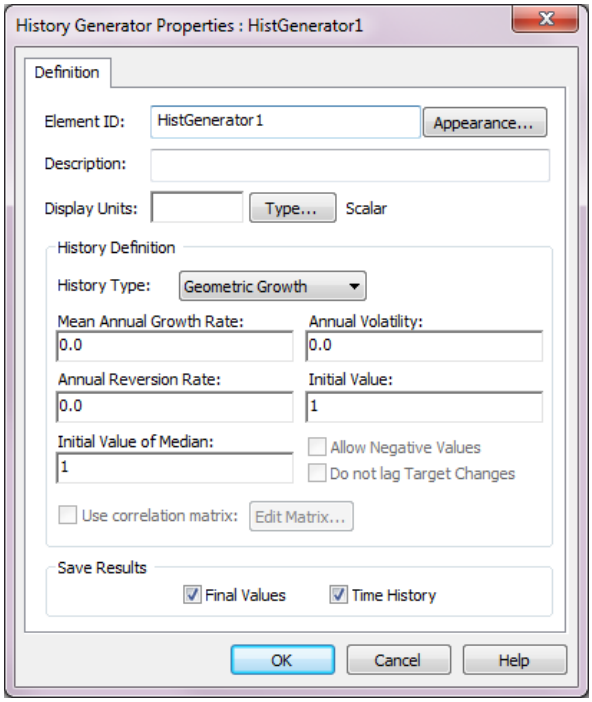
Generating Stochastic Time Histories

It is sometimes necessary to generate stochastic time histories of variables. A stochastic time history is a random time history that is generated according to a specified set of statistics. For example, for one type of stochastic time history, the "random walk", the variable can be thought of as taking successive steps, each in a random direction. The overall trend (up or down) and the size of the steps are controlled by the statistical measures specified by the user.

GoldSim provides a specialized element to generate such time histories, the History Generator. This element is primarily envisioned as a tool to allow users to simulate financial and economic variables (e.g., security prices, interest rates), but has many applications in other arenas.

The default icon for the History Generator element looks like this:

The property dialog looks like this:



Like all GoldSim elements, you first specify an **Element ID** and a **Description**.

Below the **Description** field, you specify the **Display Units** you wish to use.

To the right of the **Display Units**, you can specify the **Type**. By default, a History Generator provides a scalar output (and requires scalar inputs). However, via this button, you can also specify it to have vector (1-D array) output (and require vector inputs).

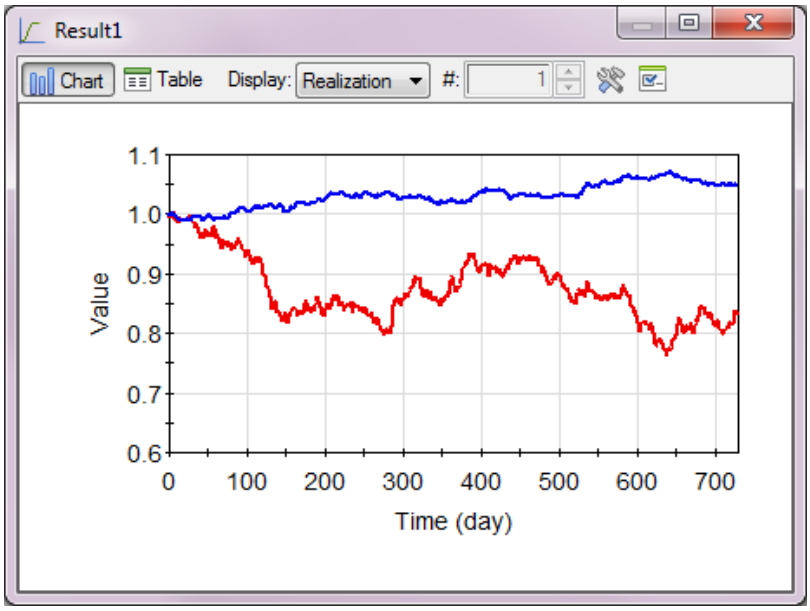
The "History Definition" portion of the History Generator dialog is used to specify the characteristics of the stochastic history. The **History Type** field is used to select the general type of stochastic history. GoldSim currently provides two options: "Geometric Growth" and "Random Walk". The details of how the History Definition fields are specified are provided below.

An example file which illustrates the use of History Generator elements (HistoryGen.gsm) can be found in the General Examples folder in your GoldSim directory.

Types of Stochastic Time Histories

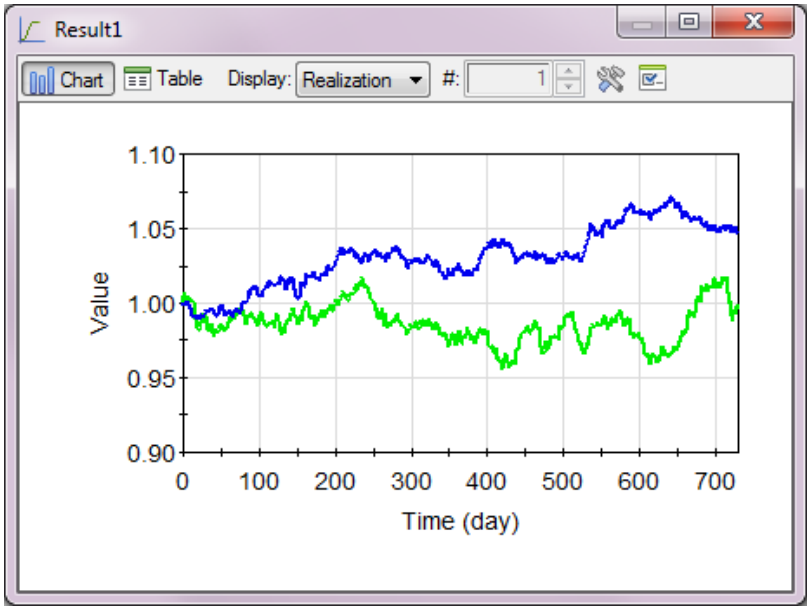
Before creating stochastic time histories using the History Generator element, it is instructive to briefly summarize the different kinds of time histories that can be generated.

Consider the stochastic time histories below:



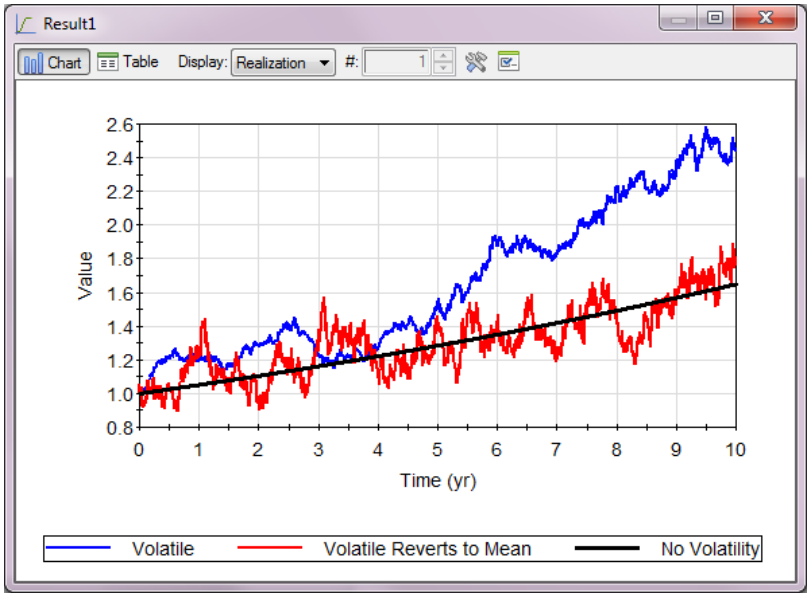
These are random histories that follow no particular trend. They simply randomly walk through time. Clearly one of the curves has larger random fluctuations than the other. For certain types of variables (e.g., stock prices), the variability of the time history as a function of time is described in terms of a volatility. The history with greater fluctuations is said to be more "volatile".

Note that the volatility is often a function of the unit of time on which it is based. In the examples shown above, the volatility increases with time (i.e., the annual volatility is greater than the daily volatility). For other types of stochastic variables, however, the volatility may stay constant with time. In the example below, the variable represented by the lower curve is target-reverting. That is, the value reverts to some underlying target (in this case, a constant value of 1). As a result, the volatility is constant in time.



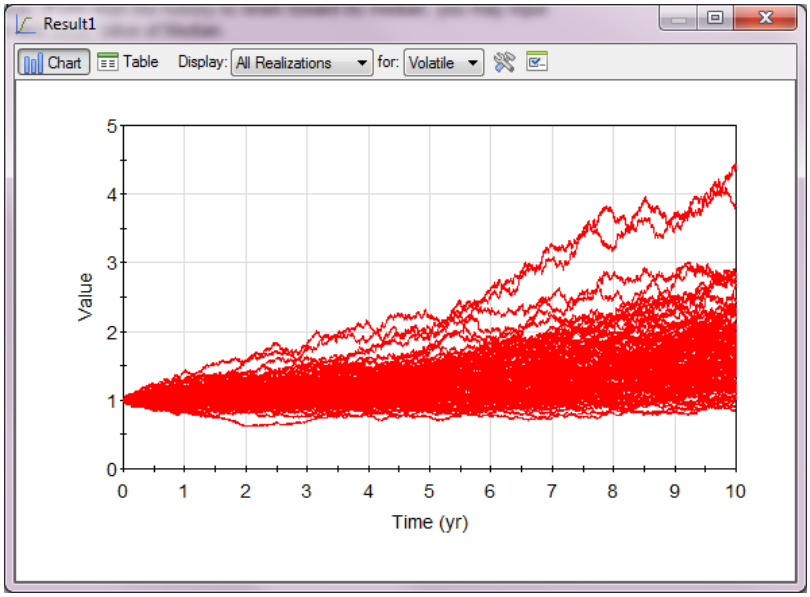
The histories in the two examples above either move randomly through time, or move randomly around some constant target. Other stochastic histories may follow a trend.

The histories below grow geometrically at a rate of 5% per year:

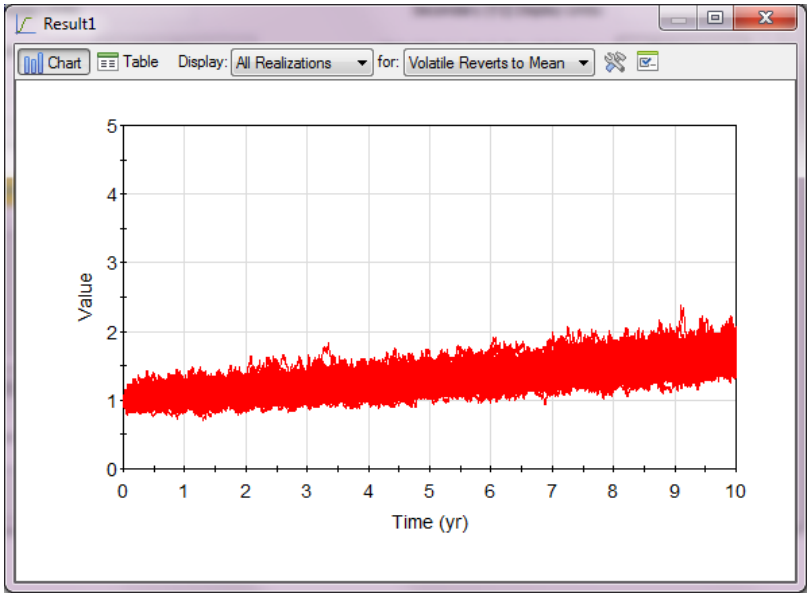


The dark smooth line is growth with no volatility. The other two lines both are volatile. In one, the volatility grows with time. In the other, the volatility stays constant.

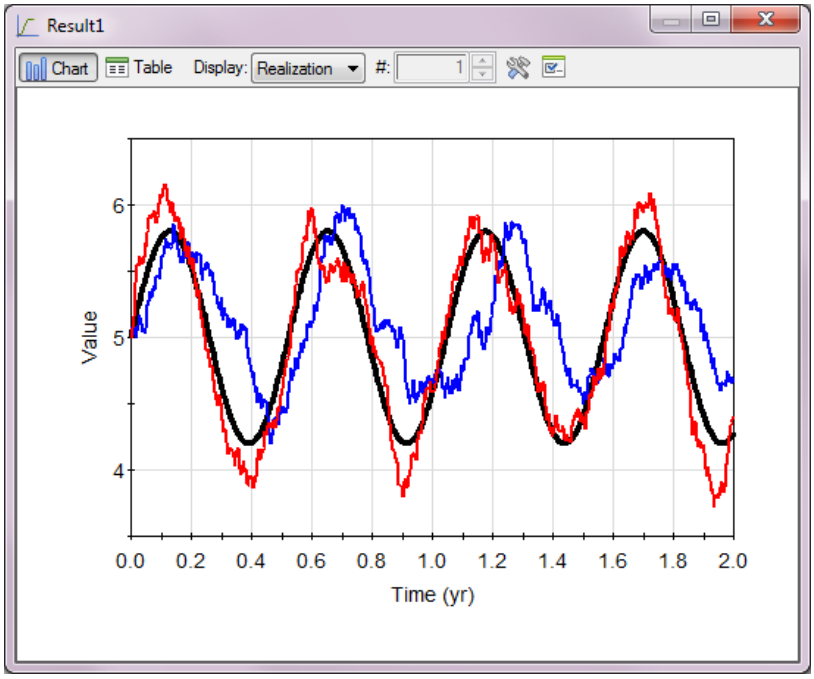
If we were to run multiple realizations of the time history whose volatility grows with time, the growth in the volatility as a function of time would be more noticeable:



Compare this to multiple realizations of the time history whose volatility stays constant with time:



Of course, the underlying trend does not need to be growth (positive or negative). The histories below follow a cyclical trend:

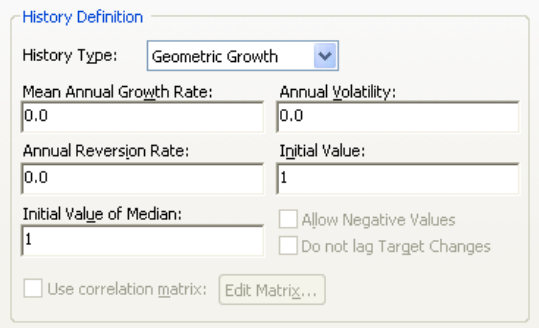


The dark smooth line is the signal with no volatility. The other two lines are both volatile. In one, history lags the underlying trend. In the other, there is no lag.

The History Generator element can be used to generate all of these types of stochastic time histories.

Generating Geometric Growth Histories

Selecting "Geometric Growth" for the **History Type** field of a History Generator element allows you to define a history that grows geometrically with time. This means that the rate of growth and the amount of volatility are proportional to the current value. Among other things, this type of history is appropriate for variables such as security prices.



In its simplest form, this type of history requires three inputs (**Mean Annual Growth Rate**, **Annual Volatility**, and **Initial Value**). The other two inputs (**Annual Reversion Rate** and **Initial Value of Median)** are only required if you wish the history to revert toward its median (and this is discussed in the next section).

Mean Annual Growth Rate. This is the mean annual *logarithmic* growth rate (μ'), where V_i is the value at time i:

$$\mu' = \frac{\sum_{i=t-n+1}^{t} \ln(\frac{V_{i+1}}{V_i})}{n}$$

Although this is a rate (i.e., time-based), because it is "hard-wired" to be an annual value, this input is dimensionless. That is, it represents the logarithmic growth rate over a year. It can be positive, negative, or zero.

If the logarithmic growth rate is measured over a different time period (e.g., monthly), this can be scaled by direct multiplication. For example a mean monthly logarithmic growth rate can be converted to a mean annual logarithmic growth rate by multiplying it by 12.

Annual Volatility. This is the standard deviation of the annual logarithmic growth rate (σ'):

$$\sigma' = \sqrt{\frac{\sum_{i=t-n+1}^{t} \left[\ln\left(\frac{V_{i+1}}{V_i} \right) - \mu'_t \right]^2}{n-1}}$$

This is a dimensionless value that must be non-negative.

Note: As pointed out above, the volatility for a geometric growth history is, by definition, dimensionless. This is different than the volatility for a random walk history, which has dimensions.

If the logarithmic volatility is measured over a different time period (e.g., monthly), and the rate of reversion to the median is zero, this can be scaled by the square root of the time period. For example, a mean monthly logarithmic volatility can be converted to a mean annual logarithmic volatility by multiplying it by $\sqrt{12}$.

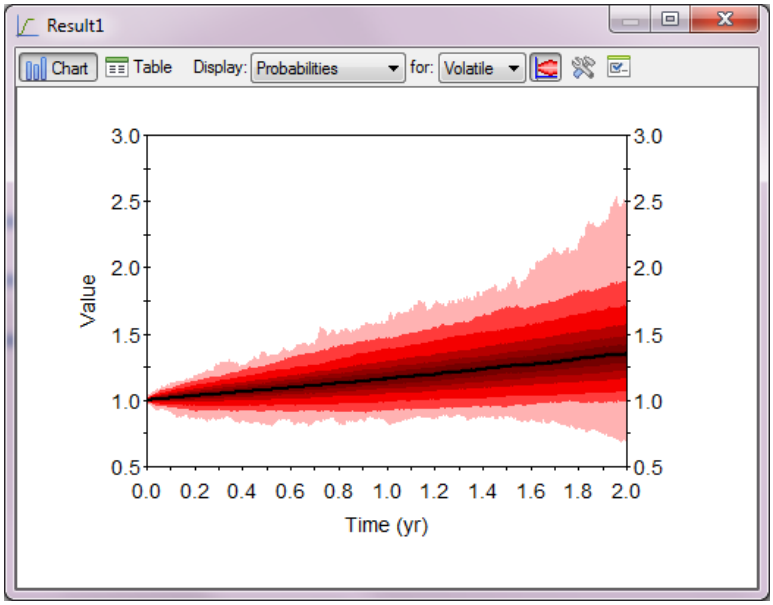
Initial Value. This is the initial value of the time history (the value at time zero). It has the same dimensions as the output. It must be a positive number.

If the **Annual Reversion Rate** is set to zero (the default), GoldSim generates successive values as follows:

$$V_{new} = V_{old} \exp\left(\mu' \Delta t + \varepsilon \sigma' \sqrt{\Delta t}\right)$$

where V_{new} is the new value, V_{old} is the previous value, Δt is the time between the two values, ε is a random standard normal value (sampled from a distribution with mean 0 and standard deviation 1), and μ' and σ' are as defined above.

For constant values of the **Mean Annual Growth Rate** and **Mean Annual Volatility**, this results in a history that (on average) grows exponentially, and whose standard deviation of the logs increases as the square root of time:



At any given time, the distribution of values is log-normal.

The logarithmic input definitions for the Growth Rate and Volatility, as used by GoldSim, are those typically used by academic financial analysts. However, it may be difficult to find data expressed using these formal definitions. Some simple approaches for converting available data to these forms are as follows:

Converting from Geometric Mean Returns. Typically, the performance of individual securities is reported as "average annual total returns". Mathematically, this actually represents the geometric mean of the annual returns over a period:

$$\mu_g = \left(\prod_{i=t-n+1}^{t} \left(\frac{V_{i+1} - V_i}{V_i} + 1 \right) \right)^{1/n} - 1$$

You can use GoldSim's built in gm2cm function to convert a geometric mean annual return to an arithmetic mean log annual return.

If the standard deviation of the average annual total returns is also available:

$$\sigma = \sqrt{\frac{\sum_{i=t-n+1}^{t} \left[\left(\frac{V_{i+1} - V_i}{V_i} \right) - \mu_a \right]^2}{n-1}},$$

$$\text{where } \mu_a = \frac{\sum_{i=t-n+1}^{t} \left(\frac{V_{i+1} - V_i}{V_i} \right)}{n}$$

then this can be used (in conjunction with the geometric mean of the returns) to compute the volatility using GoldSim's geo2vol function.

Read more: Financial Functions (page 132).

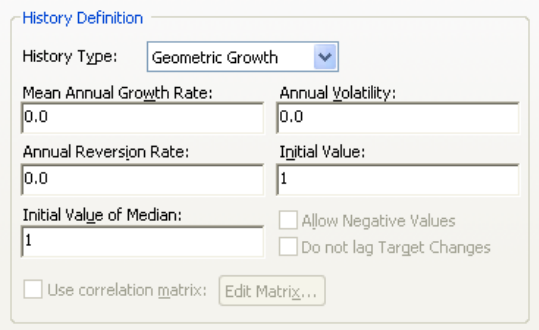
Converting from Arithmetic Mean Returns. In some cases, you may have return rate statistics reported in terms of the arithmetic mean of the annual return over a period (μ_a, defined above).

You can use GoldSim's built in ari2cm function to convert an arithmetic mean annual return (and a standard deviation) to an arithmetic mean log annual return, and the ari2vol function to convert a standard deviation (and an arithmetic mean) of the annual return to the standard deviation of the log annual return (the volatility).

Generating Geometric Growth Histories with Median Reversion

By default, the standard deviation of a geometric growth type history increases with the square root of time. In some cases, when simulating a stochastic history that increases geometrically, you may not want the standard deviation to grow in this way. Rather, you may want it to stabilize at a constant value.

You can accomplish this by forcing the stochastic history to revert towards its median value at a specified rate. To do so, you must use the **Annual Reversion Rate** and **Initial Value of Median** input fields.



Annual Reversion Rate. This is the annual fractional rate at which the history reverts towards its median. Although this is a rate, because it is "hard-wired" to an annual value, this input is dimensionless. That is, it represents the fractional rate per year. It must be non-negative.

Initial Value of Median. In most cases, the initial value of the time history is unlikely to be at the median. Hence, this is the initial value of the median of the time history (the value at time zero). It has the same dimensions as the output. It must be a positive number.

If the **Annual Reversion Rate** is non-zero, GoldSim generates successive values as follows:

$$V_{new} = V_{old} \exp\left(\mu' \Delta t + \left(1 - e^{-r\,\Delta t}\right)\left(\ln T_{old} - \ln V_{old}\right) + \varepsilon\sigma' \sqrt{\frac{1 - e^{-2\,r\,\Delta t}}{1 - e^{-2\,r\,(1yr)}}} \right)$$

where V_{new} is the new value, V_{old} is the previous value, Δt is the time (in years) between the two values, ε is a random standard normal value (sampled from a distribution with mean 0 and standard deviation 1), r is the reversion rate, and μ' and σ' are the logarithmic growth rate and volatility, respectively. T is the median (a function that grows geometrically from the specified Initial Value of Median using the mean annual growth rate).

In practical terms, specifying a non-zero Annual Reversion Rate has two effects:

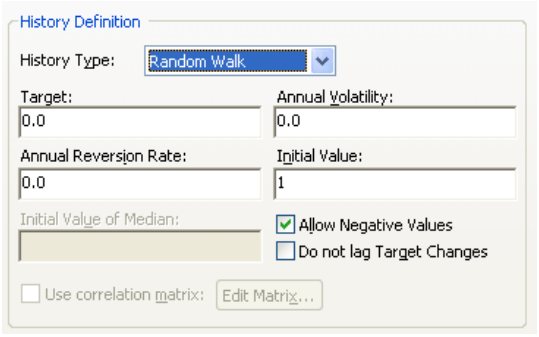
- The history tracks the median, and if the Initial Value differs from the Initial Value of Median, the history approaches the median history with a half-life of ln(2)/Annual Reversion Rate.

- The standard deviation of the logs initially grows, approaching a constant value with a similar half-life. The "steady-state" standard deviation of the log of the values stabilizes at a value of:

$$\sigma' \sqrt{\frac{1}{1-e^{-2r}}}$$

Generating Random Walk Histories

Selecting "Random Walk" for the **History Type** field of a History Generator element allows you to define a history that randomly walks through time.



In its simplest form, this type of history requires two inputs (**Annual Volatility**, and **Initial Value**). The other two inputs (**Target** and **Annual Reversion Rate**) are only required if you wish the history to track an underlying target or trend (and this is discussed in the next section).

Annual Volatility. This is the standard deviation of the annual values (σ):

$$\sigma = \sqrt{\frac{\sum_{i=t-n}^{t}[V_i - \mu)]}{n-1}}$$

where μ is the mean of the annual values:

$$\mu = \frac{\sum_{i=t-n}^{t} V_i}{n}$$

If the volatility is measured over a different time period (e.g., monthly), this can be scaled by direct multiplication. For example a mean monthly volatility can be converted to a mean annual volatility by multiplying it by $\sqrt{12}$.

It has the same dimensions as the output, and must be non-negative.

Initial Value. This is the initial value of the time history (the value at time zero). It has the same dimensions as the output and can be positive, negative or zero.

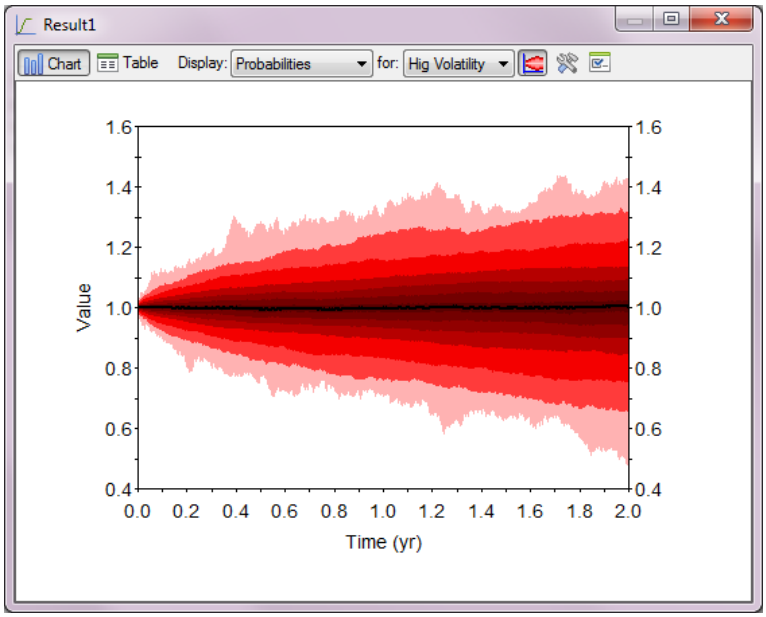
You can control whether your history can take on negative values using the **Allow Negative Values** checkbox (which defaults on). If this box is cleared, negative values are "truncated". Note that this has the effect of artificially reducing the standard deviation of the values.

If the **Annual Reversion Rate** is set to zero (the default), GoldSim generates successive values as follows:

$$V_{new} = V_{old} + \varepsilon\sigma\sqrt{\Delta t}$$

where V_{new} is the new value, V_{old} is the previous value, Δt is the time (in years) between the two values, ε is a random standard normal value (sampled from a distribution with mean 0 and standard deviation 1), and σ is the volatility.

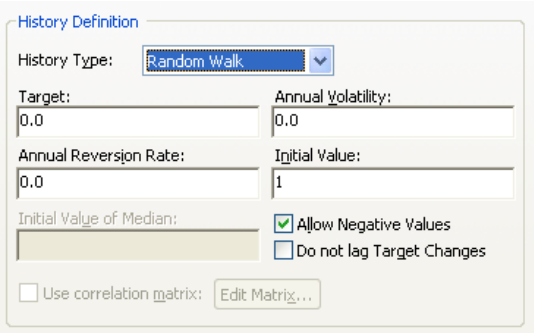
For constant values of the **Annual Volatility**, this results in a history whose standard deviation increases as the square root of time.



Generating Random Walk Histories with Target Reversion

By default, the standard deviation of a random walk type history increases with the square root of time. In some cases, when simulating such a history, you may not want the standard deviation to grow with time. Rather, you may want it to stabilize to a constant value around an underlying target.

You can accomplish this by forcing the stochastic history to revert towards a target value at a specified rate. To do so, you must use the **Annual Reversion Rate** and **Target** input fields.



Target. This is the underlying target that the history will revert to (i.e., track). It can be a constant value or a function of time. It has the same dimensions as the output. It can take on any value.

Annual Reversion Rate. This is the annual fractional rate at which the history reverts to its target. Although this is a rate, because it is "hard-wired" to an annual value, this input is dimensionless. That is, it represents the fractional rate per year. It must be non-negative.

If the **Annual Reversion Rate** is non-zero, GoldSim generates successive values as follows:

$$V_{new} = V_{old} + \left(1 - e^{-r\,\Delta t}\right)\left(T_{new} - V_{old}\right) + \varepsilon\sigma\sqrt{\frac{1 - e^{-2\,r\,\Delta t}}{1 - e^{-2\,r\,(1yr)}}}$$

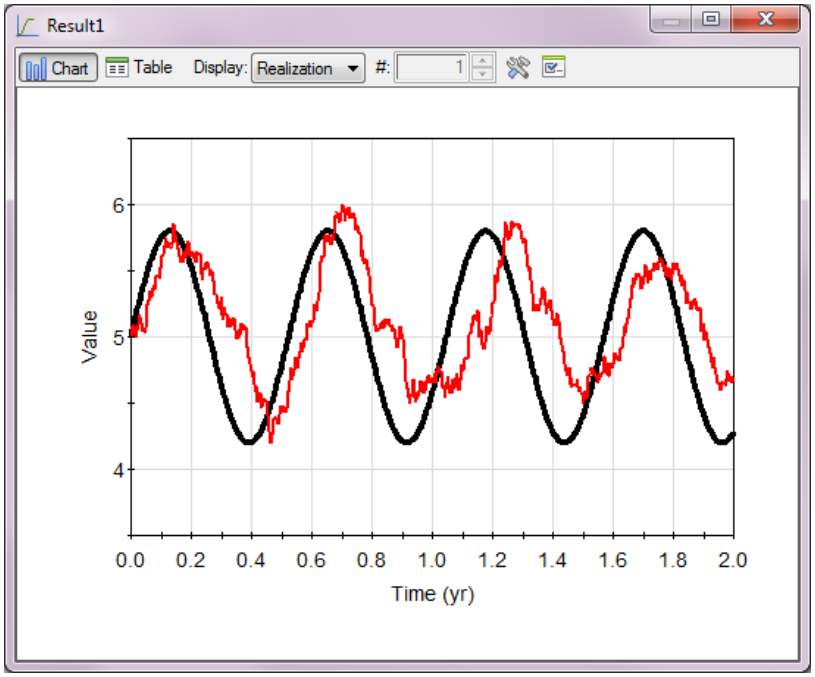
where V_{new} is the new value, V_{old} is the previous value, Δt is the time (in years) between the two values, ε is a random standard normal value (sampled from a distribution with mean 0 and standard deviation 1), r is the reversion rate, and σ is the volatility. T_{new} is the current (new) value of the underlying target (which may change with time).

In practical terms, specifying a non-zero Annual Reversion Rate has two effects:

- The history tracks the target, and if the Initial Value differs from the initial value of the Target, the history approaches the target history with a half-life of ln(2)/Annual Reversion Rate.
- The standard deviation initially grows, but eventually stabilizes to a constant value with a similar half-life. The "steady-state" standard deviation stabilizes at a value of:

$$\sigma\sqrt{\frac{1}{1-e^{-2\,r}}}$$

By default, when an Annual Reversion Rate is specified, the history actually lags the target, and as a result, the signal is "dispersed" such that the peaks and valleys of the history are smaller than that of the target:

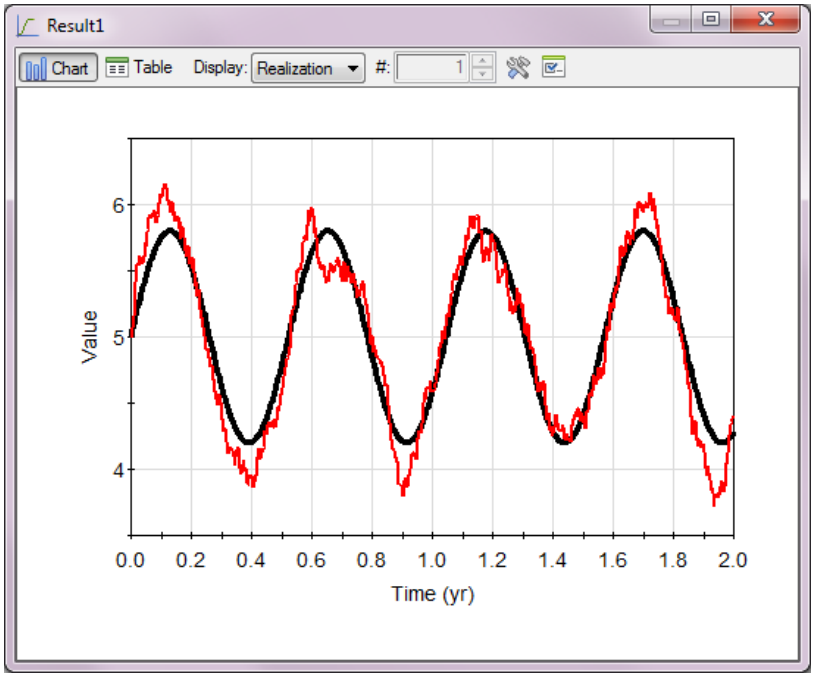


The time lag increases as the **Annual Reversion Rate** decreases.

In some cases, you may not want such a time lag. The **Do not lag Target Changes** checkbox (if checked) eliminates this lag. In this case, GoldSim generates successive values as follows:

$$V_{new} = V_{old} + \left(1 - e^{-r\,\Delta t}\right)\left(T_{old} - V_{old}\right) + \varepsilon\sigma\sqrt{\frac{1 - e^{-2\,r\,\Delta t}}{1 - e^{-2\,r\,(1yr)}}} + \left(T_{new} - T_{old}\right)$$

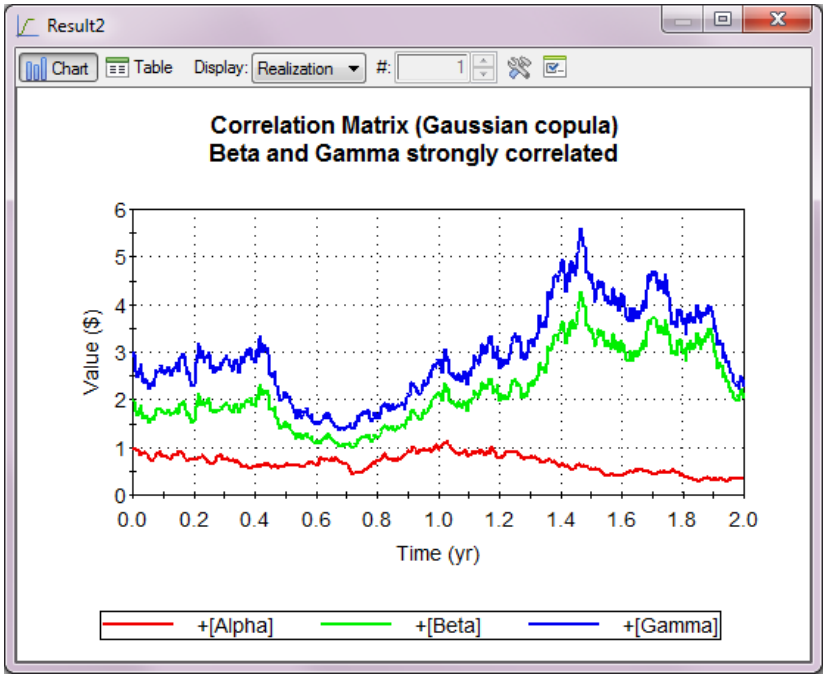
As can be seen below, this has the impact of eliminating the lag (and the dispersion):



Simulating Correlated Arrays of Stochastic Histories

A History Generator can be specified to represent a 1-D array of variables (i.e., a vector). In this case, the various input parameters must be specified as vectors.

This most common application of this is to simulate an array of correlated variables (e.g., a portfolio of investments). When variables are correlated, their random movements (as typically quantified by the volatility) are linked to a greater or lesser degree. For example, consider the three time series below:



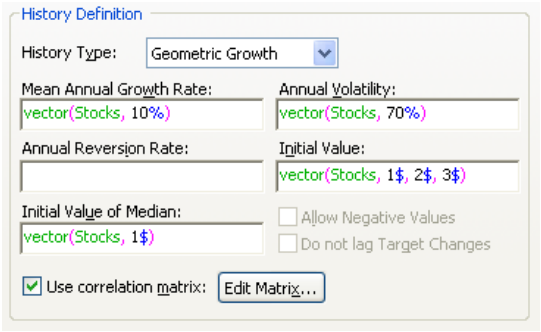
In this example, the volatile movements of Beta and Gamma are strongly correlated, and are both uncorrelated to Alpha.

The correlations between the members of an array are specified via a correlation matrix. A correlation matrix specifies the correlations between variables, and generally has the following form:

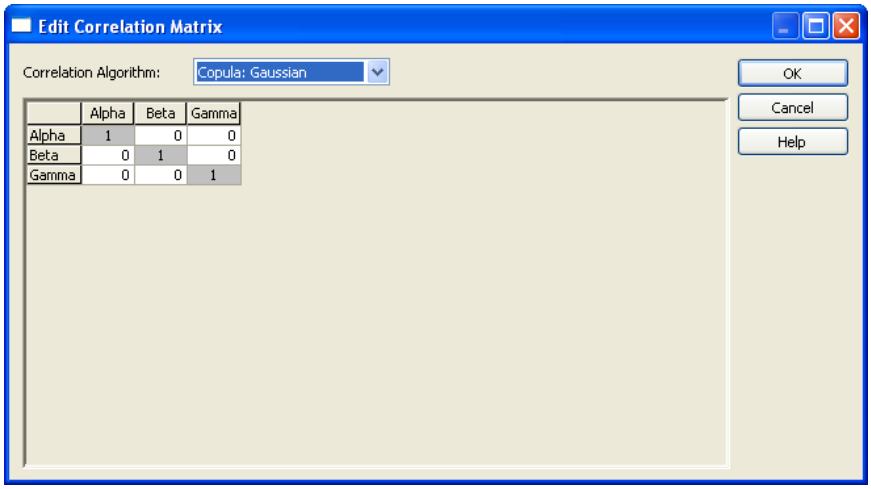
	Alpha	Beta	Gamma
Alpha	1	0.1	0.1
Beta	0.1	1	0.99
Gamma	0.1	0.99	1

Note that by definition, a correlation matrix is symmetric around its diagonal (since the cross diagonal terms define the same correlation coefficient).

If you specify that the History Generator represents a 1-D array of variables (i.e., a vector), then the **Use correlation matrix** checkbox becomes available on the History Generator dialog:



If this box is checked, the **Edit Matrix…** button becomes available. This button provides access to a dialog for specifying the correlation matrix:



By default, all off-diagonal correlation coefficients are zero. The matrix is symmetrical, so you need only define one of the cross-diagonal terms. The value represents a rank correlation coefficient, and must vary between -1 and 1. It must be a number (i.e., you cannot specify a link).

Note: When you define a correlation matrix, it is important to ensure that it is internally consistent. For example, if you specified that A was positively correlated to B, and B was positively correlated to C, but that A was negatively correlated to C, the correlation matrix would be inconsistent (since in this case, A should also be positively correlated to C). When this occurs, GoldSim will produce a fatal error message.

GoldSim provides several different algorithms for correlating the members of the vector. These are selected from the **Correlation Algorithm** drop-list at the top of the dialog. The various correlations algorithms are discussed in detail in Appendix B.

Using Resources

One of the advanced features in GoldSim is the ability to create and interact with Resources. A ***Resource*** is something that has a limited supply (e.g., spare parts, fuel, skilled personnel, money) and is required in order for elements of the modeled system to carry out certain actions.

For example, in order for a manufacturing process to proceed, perhaps two different parts and a skilled operator must be available. This could be modeled by defining an Event Delay (to represent the process) with three Resource requirements that must be met in order for the element to process the event.

Alternatively, in order for a project task to continue to operate, perhaps a certain amount of cash must be available. This could be modeled by defining a Conditional Container (to represent the task) with a single Resource requirement (a spend rate) that must be met in order for the task to continue (i.e, for the Container to stay active).

Only certain elements in GoldSim can interact with Resources (two of these are mentioned in the examples above: an Event Delay and a Conditional Container). Elements can consume, borrow, and in some cases, generate Resources.

Using Resources involves three different steps:

- You must define a Resource Type (e.g., Gasoline, Electricians, Pump Motors) and specify its characteristics (discrete or continuous).
- You must specify where the ***Stores*** for that particular Resource exist in your model. Stores represent stockpiles or places where the actual Resource (e.g., parts, personnel) is stored or located when not being used. That is, Resource Stores can be thought of as having physical locations in the system you are modeling.
- You must specify how particular elements in the model interact with (consume, borrow from, or add to) the Resource Stores.

The sections below discuss the use of Resources in detail.

A simple example which illustrates the use of Resources (Resources.gsm) can be found in the General Examples folder of your GoldSim directory.

Defining Resource Types and Creating Resource Stores

A ***Resource*** is something that has a limited supply (e.g., spare parts, fuel, skilled personnel, money) and is required in order for elements of the modeled system to carry out certain actions.

Creating a Resource requires two distinct steps:

1. First you must define the type of Resource (e.g., Gasoline, Electricians, Pump Motors) and specify its characteristics (discrete or continuous).
2. Then you must specify where the ***Stores*** for that particular Resource exist in your model.

Stores represent stockpiles or places where the actual Resource (e.g., parts, personnel) is stored or located when not being used. That is, Resource Stores can be thought of as having physical locations in the system you are modeling.

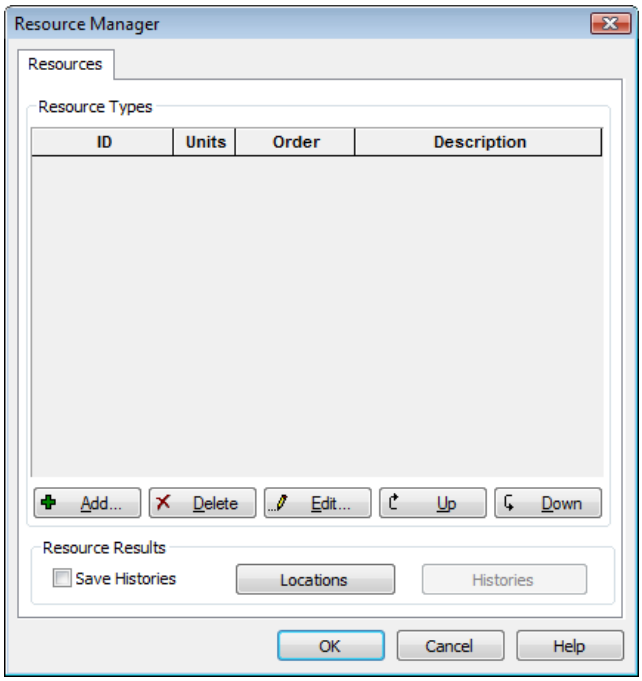
There are two types of Stores that can be created: ***Global Stores*** and ***Local Stores***. Global Stores are available to all elements in your model. Local Stores are associated with specific Containers, and are only available to elements inside the Container (or in the case of a Conditional Container, to the Container itself).

A Resource Type can have multiple Stores in a model. For example, you could define multiple Local Stores, or perhaps a Global Store and several Local Stores.

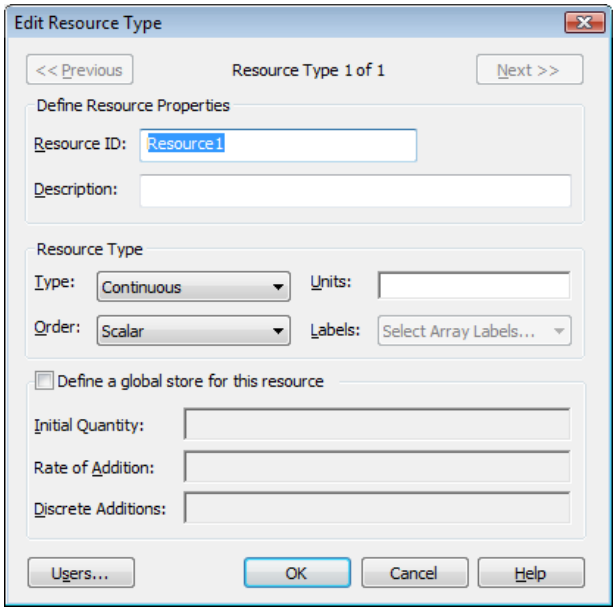
These two steps are described in the sections below.

Defining and Editing Resource Types

You define or edit Resource types by using the Resource Manager, which is accessed via the main menu (**Model | Resources…**) or by pressing **F8**. The following dialog will be displayed:



Pressing the **Add…** button displays the following dialog for defining a new Resource Type:



Each Resource Type requires a unique **Resource ID**. Resource IDs follow the same rules as elements names (e.g., no spaces, letters, only numbers and the underscore character can be used; cannot start with a number). You should also provide a brief **Description**.

A Resource has the following attributes:

Type: This can be "Continuous" or "Items". Continuous should be used for things that are not discrete (e.g., fuel). Items should be specified for things that are discrete (e.g., parts, personnel).

Units: If the Resource is specified as Continuous, in most cases you will want to specify units (e.g., m^3, kg). If the Resource is specified as Items, units are not applicable.

Order: A Resource can be defined as a scalar or a vector. Defining a Resource as a vector is most useful, for example, if you have a large number of indidual parts. Defining these as a vector is much easier than defining a separate Resource for each part.

Read more: Using Vectors and Matrices (page 726).

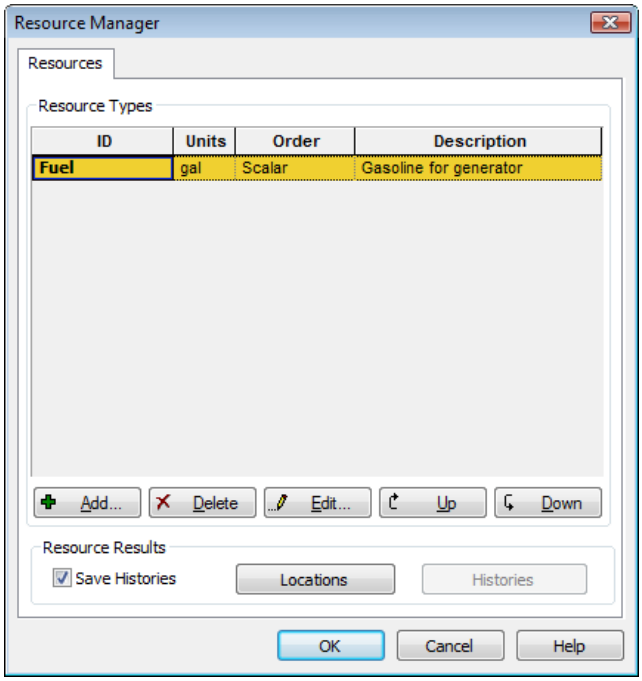
Labels: If you define the Resource as a vector, you need to specify a set of Array Labels from the Labels drop-list.

Read more: Understanding Array Labels (page 727).

Note: The bottom part of the dialog provides an option to create a "Global Store" of this particular Resource. This is discussed in detail in the next section.

If multiple Resource Types are defined, you can move between them using the **Previous** and **Next** buttons.

After you have finished defining the Resource, press the **OK** button to return to the Resource Manager dialog:

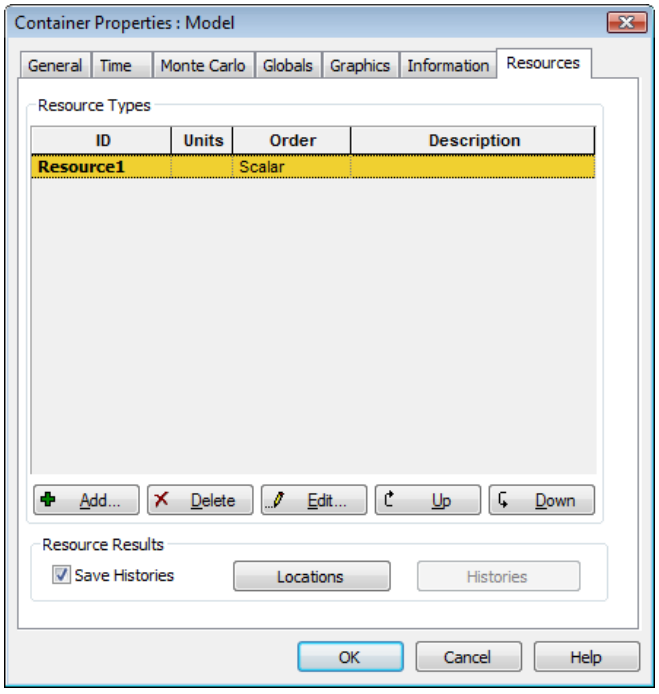


You can delete and edit existing Resource Types using the **Delete** and **Edit** buttons, respectively.

Note: If a selected Resource Type is being used by an element in the model, or has a Local Store defined, GoldSim will display an error message if you try to delete it. You can only delete Resource Types that are not being used.

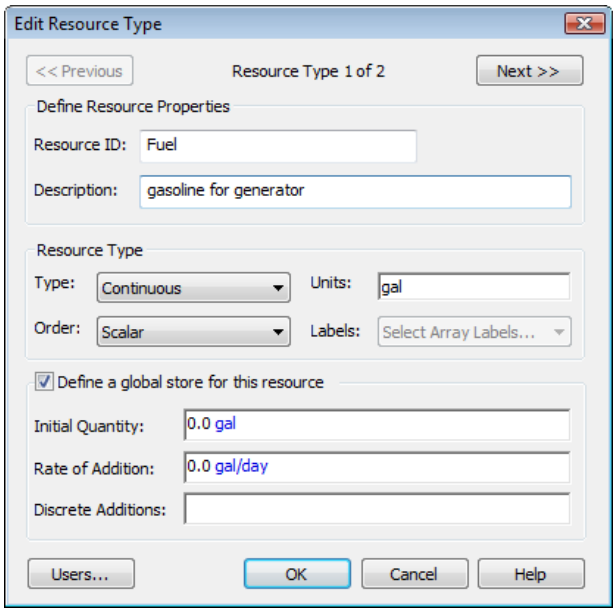
The **Up** and **Down** buttons move Resources up and down the list (this is simply cosmetic; the order in which they are listed has no impact).

Note that you can also access the Resource Manager by right-clicking in the graphics pane at the top level of the model and selecting Properties. The Resource Manager is available as a tab on the Model Properties dialog:



Creating and Editing Global Resource Stores

Global Resource Stores are available to all elements in your model. Global Stores are created within the Resource Manager (accessed via **Model| Resources…** or by pressing **F8**).When you **Add** or **Edit** a Resource Type, you have the option to define a Global Store for that Resource by checking **Define a Global Store for this Resource**:



This creates a Global Store and provides access to the three input fields to define the Store: the **Initial Quantity**, the **Rate of Addition** and **Discrete Additions**.

The Initial Quantity must have the same attributes (order and dimensions) as the Resource Type.

Note: The Initial Quantity must be a number or a link from a static variable (e.g., a constant Data element or a Stochastic).

The Rate of Addition allows you to add to your Resource Store in a continuous manner. It is not provided if the Resource Type is "Items". It must have the same order as the Resource Type, but the dimensions must represent a rate of change (e.g., if the Resource Type has units of mass, the Rate of Addition must have units of mass per time). The Rate of Addition must be entered as a non-negative value. A negative value for this input during a simulation will result in a fatal error.

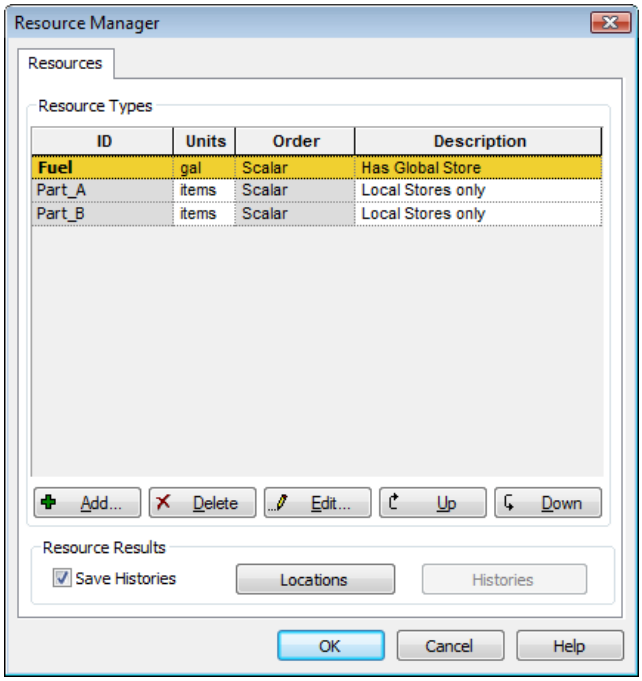
Note: The specified Rate of Addition represents a constant rate over the next timestep. Hence, if the Rate of Addition was defined as "if(time > 10 day, 2 m^3/day, 1 m^3/day)", and you were using a 1 day timestep, the rate would not actually change to 2 m^3/day until time = 11 days. That is, since at time = 10 days, the if statement indicates that the rate is equal to 1 m^3/day, GoldSim would assume that the rate was equal to 1 m^3/day between 10 days and 11 days. If you wanted the rate to change at 10 days, you would write the if statement as "if(time >= 10 day, 2 m^3/day, 1 m^3/day)".

In addition to a continuous Rate of Addition, Resource Stores can also accept discrete changes, so that your Resource Store can change in a discrete manner. The Discrete Additions field only accepts discrete change signals (e.g., the output of a Discrete Change element). The discrete change signal must have a non-negative value. However, it can have either an Add instruction or a Replace instruction.

The **Users** button summarizes the properties of the Store, and lists all the elements that directly use this Store.

Read more: Summarizing Resource Locations and Users (page 798).

After you have created a Global Store, you will notice that Resource Types that have Global Stores are marked in bold in the Resource Manager:



The **Locations** button summarizes the properties of all of the Resource Stores in the model, as well as all of the elements that directly use those Stores.

Read more: Summarizing Resource Locations and Users (page 798).

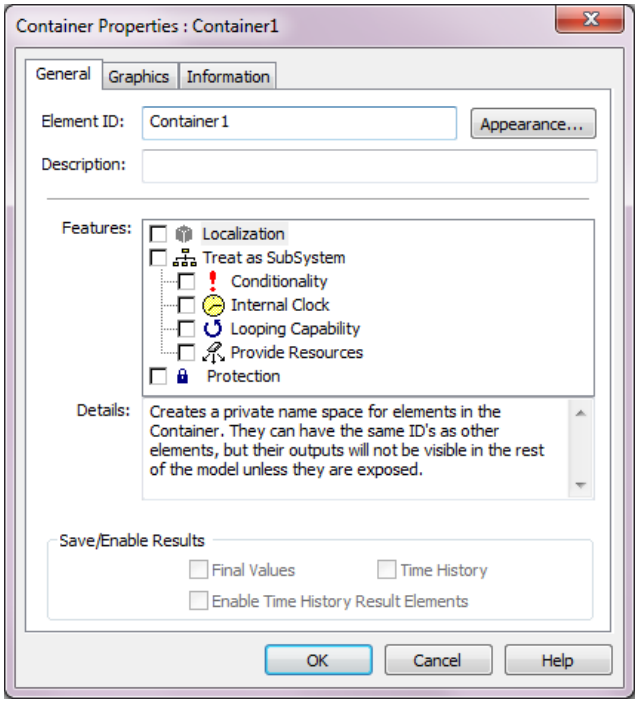
If **Save Histories** is checked, GoldSim saves time histories of all of the Resource Stores in the model and users of those Stores. After the model is run, these histories can be viewed via the **Histories** button.

Read more: Viewing Resource Results (page 802).

Creating and Editing Local Resource Stores

Local Resource Stores are associated with specific Containers, and are only available to elements inside the Container (or in the case of a Conditional Container, to the Container itself).

Local Stores are created via the Container dialog:

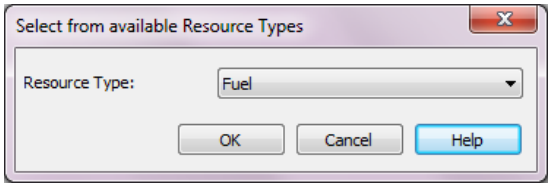


Selecting the **Provide Resources** checkbox adds a **Resources** tab to the Container that allows you to create Local Stores.

Warning: When you provide Resources to a Container, the **Treat as SubSystem** feature is also automatically selected (and cannot be deselected unless you first turn off **Provide Resources**). That is, a Container with a Resource Store, by definition, is treated as a SubSystem, and this puts certain limitations on how these Containers can be used.

Read more: Treating a Container as a SubSystem (page 139).

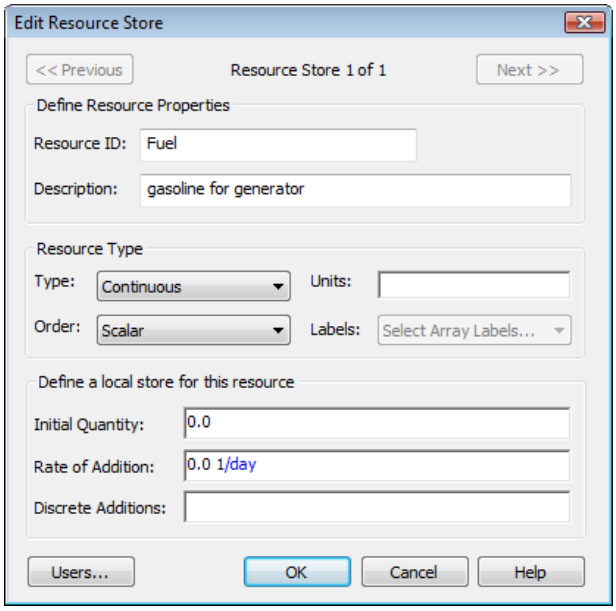
Selecting the **Resources** tab and pressing the **Add** button displays the following dialog for selecting the Resource Type for which you want to create a Local Store:



All *available* Resource Types will be listed. Note that a defined Resource Type is available as long as a Store for that Type has not already been created in the Container. That is, there can only exist a single Store of each Type in any given Container.

Note: If there is only one Resource Type in the model and it is available for use in the Container, the dialog above will be skipped and the single Resource Type will be automatically selected.

After selecting the Resource Type, the following dialog will be displayed:



This creates a Local Store and provides access to the three input fields to define the Store: the **Initial Quantity**, the **Rate of Addition** and **Discrete Additions**.

The Initial Quantity must have the same attributes (order and dimensions) as the Resource Type.

Note: The Initial Quantity must be a number or a link from a static variable (e.g., a constant Data element or a Stochastic).

The Rate of Addition allows you to add to your Resource Store in a continuous manner. It is not provided if the Resource Type is "Items". It must have the same order as the Resource Type, but the dimensions must represent a rate of change (e.g., if the Resource Type has units of mass, the Rate of Addition must have units of mass per time). The Rate of Addition must be entered as a non-negative value. A negative value for this input during a simulation will result in a fatal error.

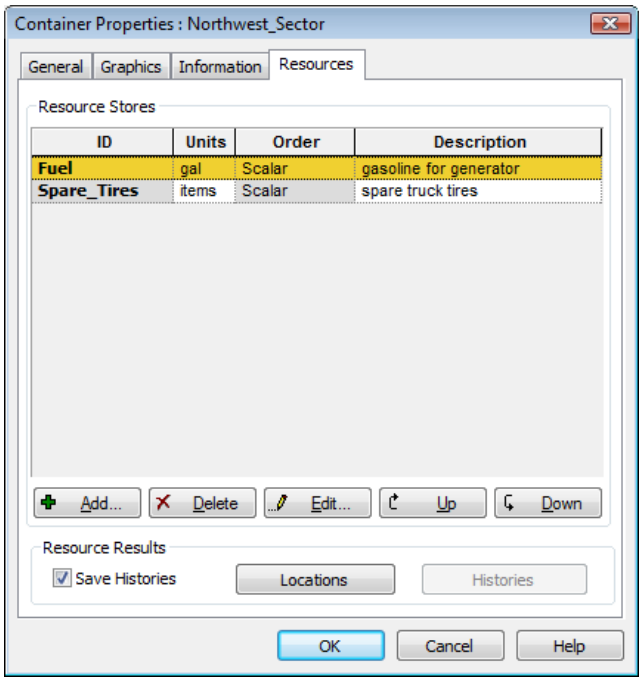
Note: The specified Rate of Addition represents a constant rate over the next timestep. Hence, if the Rate of Addition was defined as "if(time > 10 day, 2 m^3/day, 1 m^3/day)", and you were using a 1 day timestep, the rate would not actually change to 2 m^3/day until time = 11 days. That is, since at time = 10 days, the if statement indicates that the rate is equal to 1 m^3/day, GoldSim would assume that the rate was equal to 1 m^3/day between 10 days and 11 days. If you wanted the rate to change at 10 days, you would write the if statement as "if(time >= 10 day, 2 m^3/day, 1 m^3/day)".

In addition to a continuous Rate of Addition, Resource Stores can also accept discrete changes, so that your Resource Store can change in a discrete manner. The Discrete Additions field only accepts discrete change signals (e.g., the output of a Discrete Change element). The discrete change signal must have a non-negative value. However, it can have either an Add instruction or a Replace instruction.

The **Users** button summarizes the properties of the Store, and lists all the elements that directly use this Store.

Read more: Summarizing Resource Locations and Users (page 798).

After you have added one or more Local Stores, the **Resource** tab for a Container shows all of the Resource Stores associated with the Container:



You can delete and edit existing Resource Stores using the **Delete** and **Edit** buttons, respectively.

Note: If a Resource Store is being used and you delete it, GoldSim will allow this, but GoldSim will display an error when you try to run the model.

The **Up** and **Down** buttons move Resources up and down the list (this is simply cosmetic; the order in which they are listed has no impact).

The **Locations** button summarizes the properties of all Stores that exist in the Container, as well as all of the elements that directly use those Stores.

Read more: Summarizing Resource Locations and Users (page 798).

If **Save Histories** is checked, GoldSim saves time histories of all of the Resource Stores in the model and users of those Stores. After the model is run, these histories can be viewed via the **Histories** button.

Read more: Viewing Resource Results (page 802).

Interacting with Resources

Once you have defined some Resources and created one or more Resource Stores, you can specify ways in which elements in your model interact with those Resource Stores.

Elements can interact with a Resource Store in three different ways:

- An element can *Spend* (consume) a Resource from the Store.

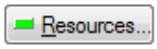
- An element can *Borrow* a Resource from the Store for a particular amount of time and then return it.
- An element can *Deposit* (generate*)* a Resource into a Store.

These interactions can be discrete or continuous in nature. As a result, there are actually five different kinds of possible Resource interactions:

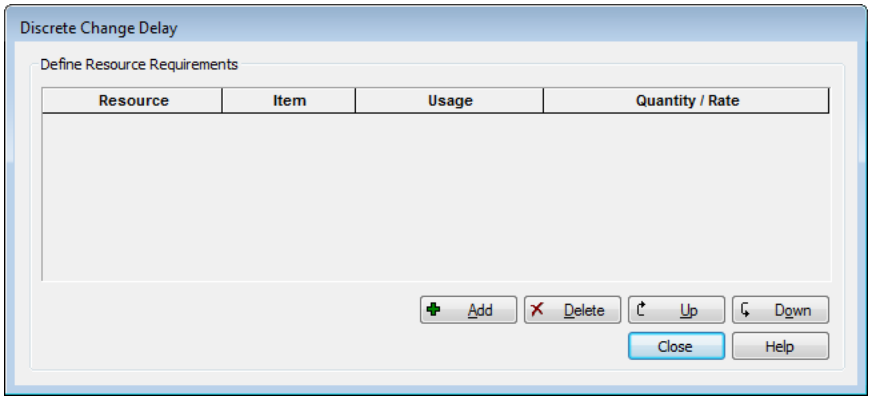
- You can Spend a discrete amount of a Resource;
- You can Borrow a discrete amount of a Resource;
- You can Deposit a discrete amount of a Resource;
- You can Spend a Resource at a specified rate; and
- You can Deposit a Resource at a specified rate.

Only certain kinds of elements in GoldSim can interact with Resources, and not all five types of interactions may be available to a particular element.

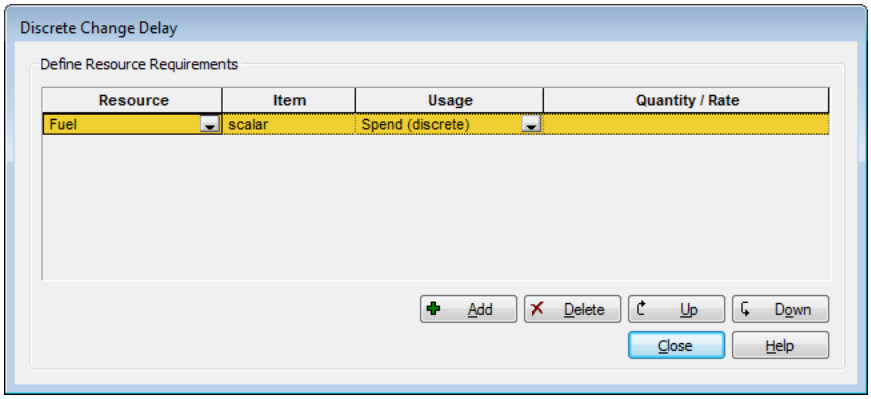
Whenever an element does interact with a Resource Store, it does so through a **Resources…** button:



Depending on the context, this button is either available on the main dialog for the element or within a Trigger dialog for the element. When you press the button, the following dialog is displayed:



A Resource Requirement (or more generally, an interaction) is created by pressing the **Add** button. When you do this, GoldSim will add a row to the table of Resource Requirements:



You must then edit the row to define the Resource Requirement. The **Resource** column is a drop-list that contains all Resource Stores that are *available* to the element.

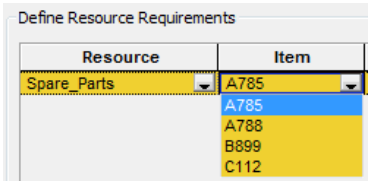
What Stores are available to an element is a function of where the element is located in the model and the defined hierarchy of Local and Global Stores. The following rules are applied:

- An element can only interact with Resource Stores that are above it or at the same level within its Container path. For example, if the element is located within Container1/Container2 (i.e., Container2 which is inside Container1), the element can only interact with Local Stores associated with Container1 or Container2, and Global Stores. The element would not be able to interact with any Local Stores associated with Container3 (a Container that was at a parallel level to Container1).
- If a Resource Type has multiple Stores within a particular Container path, the element can only access the Store that is closest to it. For example, if the element is located within Container1/Container2 (i.e., Container2 which is inside Container1), and Container1 and Container2 have Local Stores of a particular Resource Type, and there is also a Global Store of that Type, the element can only access the Store in Container2, *even if that Store becomes empty during the simulation*. If the Store in Container2 was deleted, it could then only access the Store in Container1. Only if the Store in Container1 was also deleted could it access the Global Store.

In some cases, if one Resource Store is exhausted, you may want to move Resources from another Store. This can be done, but you must manually program how and when this takes place.

Read more: Moving Resources in a Model (page 795).

The **Item** column is not editable if the Resource is a scalar. However, if it is a vector, the column provides a drop-list for you to select the item from the vector for which the Resource Requirement is defined:



The **Usage** column specifies the type of interaction (e.g., Spend, Borrow, etc.). As noted above, there are five possible types of interactions, but depending on the element type and the Resource Type, not all of these will be available (e.g., a Spend Rate is not available for Resource Types that are discrete items).

The **Quantity/Rate** column is an input field in which you specify a value that is appropriate for the particular type of Usage (e.g., discrete usages require quantities, continuous usages require rates). You can enter a value or an expression (or links to other elements). It can also be a function of time. However, the dimensions must be consistent with the specified Resource Type. Within this field, you can also reference several locally available properties pertaining to the amount of Resource that is available, or has been spent or borrowed.

Read more: Referencing Resource Availability and Use in Input Expressions (page 794).

Note: If a Resource becomes exhausted (due to Spend Rate) between scheduled updates (timesteps), GoldSim will insert an unscheduled update to accurately capture this. If you have disabled unscheduled updates (an advanced timestepping option), such that GoldSim cannot insert an unscheduled update when the Resource becomes exhausted, a fatal error will be displayed.

Read more: Understanding Timestepping in GoldSim (page 415).

A Resource Requirement can be removed by pressing the **Delete** button. The **Up** and **Down** buttons move Resource Requirements up and down the list (this is simply cosmetic; the order in which they are listed has no impact).

Note: If you have multiple Resource Requirements, they are only applied when they can all be met. For example, if you had a Spend requirement of 3 of PartB and 2 of PartC, and the Stores had enough of PartB, but not of PartC, neither PartB or PartC would be consumed (until both were available in sufficient quantities). In addition, all Spend requirements must first be met before any Deposit interactions are carried out.

Note: You can create multiple Resource interactions for the same Resource Store. For example, you might want to Borrow a certain quantity, and Consume (Spend) a different quantity.

The types of elements that can interact with Resources are summarized in the next section.

Elements That Can Interact With Resources

Only certain kinds of elements in GoldSim can interact with Resources. The table below summarizes which elements can use Resources, and how they can use them:

Element	Resource Store Interaction	Allowed Usage	Comments
Triggered Event	Interacts when triggered.	Spend (discrete) Deposit (discrete)	If triggered with a Spend Requirement and Resource is not available, trigger is held (it waits) for Resource.
Event Delay	Interacts when triggered. Borrowed items are returned when signal is released.	Spend (discrete) Borrow (discrete) Deposit (discrete)	If Requirement is Spend or Borrow, and Resource is not available, the signal is added to the queue for the element.
Discrete Change Delay	Interacts when discrete change signal is received. Borrowed items are returned when signal is released.	Spend (discrete) Borrow (discrete) Deposit (discrete)	If Requirement is Spend or Borrow, and Resource is not available, the signal is added to the queue for the element.

Element	Resource Store Interaction	Allowed Usage	Comments
Conditional Container	Interacts when Activation is triggered. Borrowed items are returned when Container is deactivated.	Spend (discrete) Borrow (discrete) Deposit (discrete)	If triggered with a Spend or Borrow Requirement and Resource is not available, trigger is held (it waits) for Resource.
	Interacts while Container is Active	Spend Rate Deposit Rate	While Container is active, spends or borrows at specified rate. If Spend Rate is specified and Resource is not available, the Container deactivates until the Resource becomes available.

Specialized elements in the GoldSim Reliability Module can also interact with Resources.

Read more: Specifying Resources for a Triggered Event (page 337); Specifying Resources for an Event Delay (page 350); Specifying Resources for a Discrete Change Delay (page 362); Specifying Resources for a Conditional Container (page 849).

Controlling How Resources are Allocated Amongst Competing Requirements

In some cases, a Resource Store may have competing requirements. For example, perhaps 10 kg of a particular Resource exists in a Store, and two Triggered Events are triggered at the same time, each one having a Discrete Spend Resource Requirement of 6 kg. How does GoldSim decide which element receives the requested Resource?

GoldSim uses the the ***causality sequence*** to determine how Resources are allocated amongst competing requirements.

When you build a model, GoldSim automatically *sequences* the elements in the order that they must be computed. For example if A was a function of B, and B was a function of C, C would be sequenced first, followed by B, followed by A. This is referred to as the causality sequence. Although in this simple example, the sequence is obvious, for complex models with looping logic, the causality sequence may not be readily apparent. The causality sequence can be viewed by selecting **Model|Causality Sequence** from the main menu.

Read more: The Causality Sequence and Element Updating (page 311).

GoldSim also provides a way to manipulate the causality sequence so that elements are computed in a specified order.

Read more: Addressing Ambiguous Causality Sequences (page 911).

Referencing Resource Availability and Use in Input Expressions

In some situations, when defining a Resource Requirement **Quantity/Rate**, or any other property of an element that is interacting with Resources (e.g., a trigger), you may want to make the input a function of the current state of one or more Resource Stores.

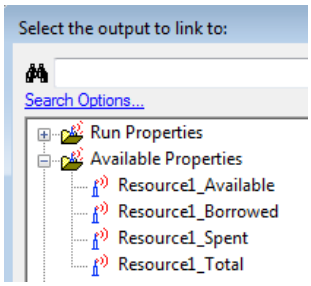
GoldSim provides four locally available properties for each Resource Store available to an element:

- ***ResourceName*_Available** (e.g., Fuel_Available): The quantity of the specified Resource Store that is currently available.

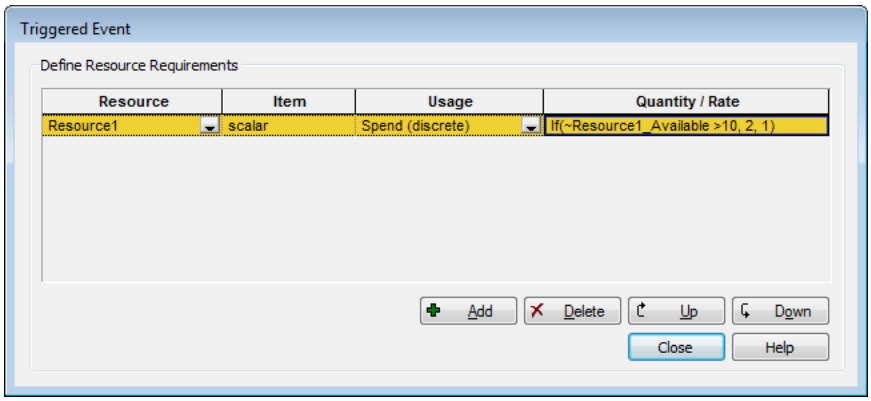
- ***ResourceName*_Borrowed** (e.g., Fuel_Borrowed): The quantity of the specified Resource Store that is currently borrowed.
- ***ResourceName*_Spent** (e.g., Fuel_Spent): The quantity of the specified Resource Store that has been spent.
- ***ResourceName*_Total** (e.g., Fuel_Total): The sum of *ResourceName*_Available and *ResourceName*_Borrowed.

Read more: Understanding Locally Available Properties (page 750).

Like all locally available properties, these are accessed via the Insert Link dialog, in this case by first right-clicking inside an input field of an element that can use Resources, and then expanding the Available Properties folder (displaying all of the locally available properties that can be accessed from that location):



Like all locally available properties, these appear in input fields with the prefix "~" (e.g., ~Fuel_Available):



Moving Resources in a Model

Because 1) an element can only interact with Resource Stores that are above it or at the same level within its Container path, and 2) if a Resource Type has multiple Stores within a particular Container path, the element can only access the Store that is closest to it, situations might arise where you may want to move Resources between Stores.

For example, if you had a Local Store of "Fuel", as well as a Global Store of "Fuel", if the Local Store was exhausted and the Global Store still had available Resources, elements would not be able to acess the Global Store in order to meet their requirements. As long as a Local Store exists that is available to the elements, this would be the only Store that could be accessed by those elements. To access the Global Store, therefore, you would need to *move* Resources from the Global Store to the Local Store.

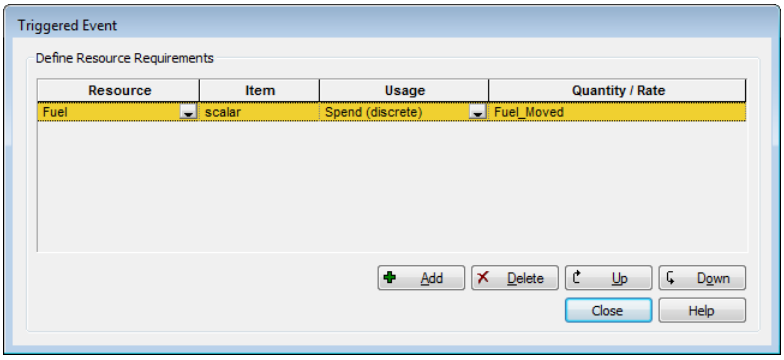
This can be done, but you must manually program how and when this takes place. The recommended way to do this is to use a combination of a Triggered

Event and a Discrete Change element to remove the Resource from one Store (e.g., the Global Store) and deposit it in the other Store (e.g., the Local Store).

The specific steps involved in doing so are as follows:

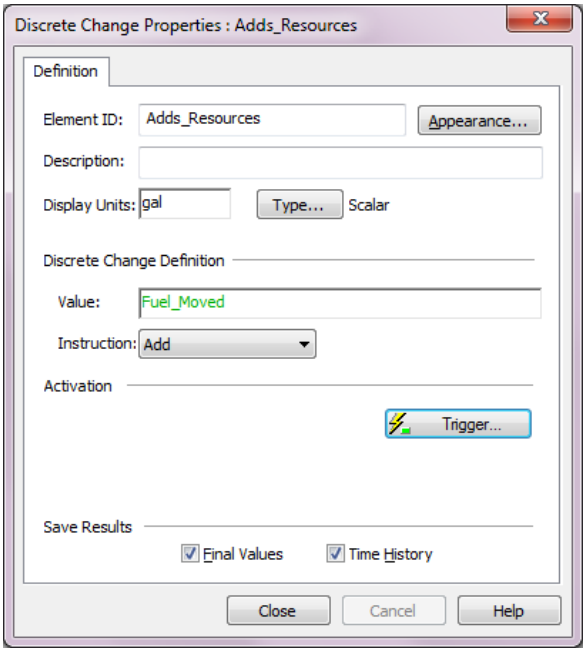
1. Add a Triggered Event element at a location in the model such that it can access the Store from which you want to remove Resources (e.g., if you are moving from a Global Store to a Local Store, the Triggered Event element should be above the scope of the Local Store).

2. The Triggered Event element should have a Spend (Discrete) Resource Requirement that pulls Resources from the first Resource Store:

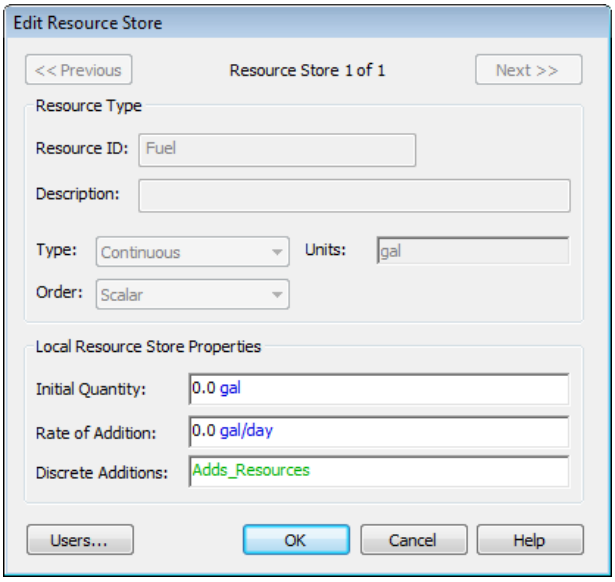


Read more: Specifying Resources for a Triggered Event (page 337).

3. Add a Discrete Change element that is triggered by the Triggered Event. This element could be located either at the same location as the Triggered Event or at the location of the Store to which the Resources are being moved. The key point is that the Value for the Discrete Change should be the same as the Quantity that was specified for the Resource Requirement for the Triggered Event:

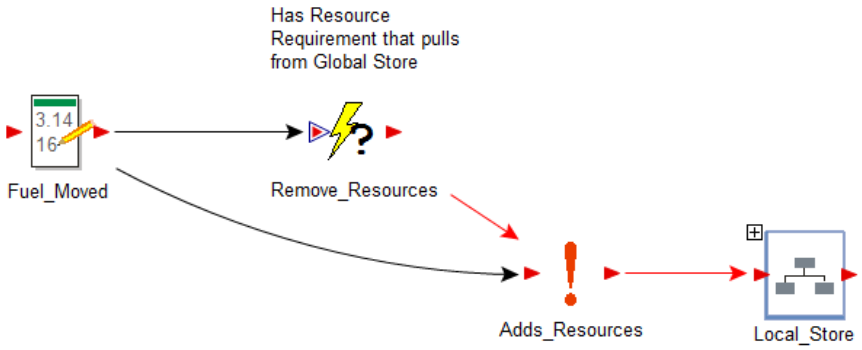


4. Within the Store to which the Resource is being moved, specify a Discrete Addition that is the output from the Discrete Change element:



Read more: Generating Resources (page 797).

A view of this structure is shown below:



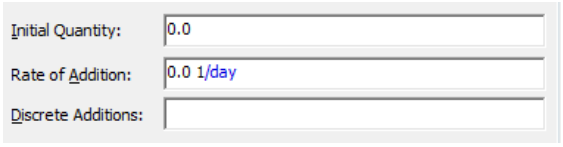
Note: If the process of moving the Resource takes time, you could represent this by adding a Discrete Change Delay after the Discrete Change element.

Generating Resources

When using Resources in your model, you will define a number of events or processes that consume those Resources. Generally, you will specify an Initial Quantity when defining the Resource Store.

In some cases, however, you may want to add Resources to a Store during a simulation. GoldSim provides two ways to do this:

- Within the Store, add a Rate of Addition or Discrete Additions.



- Specify a Deposit Rate or a Discrete Deposit when defining a Resource Interaction.

The first of these two options is perhaps the most powerful, as you have great flexibility in how you can add resources. Note that the **Discrete Additions** field also accepts discrete change signals that have a Replace instruction. This, for example, would allow you to reset a Store to a pre-specified level (e.g., its capacity) regardless of its current level.

Read more: Creating and Editing Global Resource Stores (page 785); Creating and Editing Local Resource Stores (page 787).

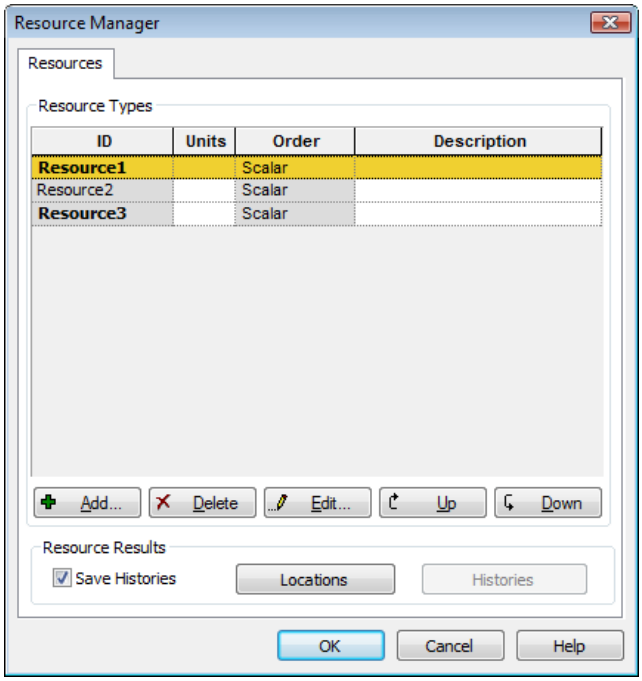
All elements that can interact with a Resource Store can add (deposit) rather than spend Resources.

Read more: Elements that Can Interact With Resources (page 793).

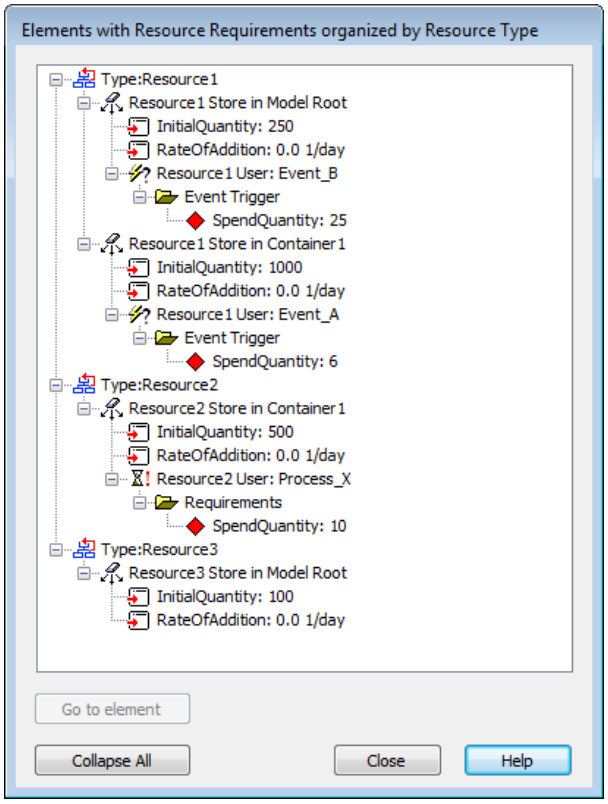
Summarizing Resource Locations and Users

After you have added a number of Resource Stores and elements that interact with those Stores to your model, it is often useful to be able to summarize the locations and users of the various Resources that exist in your model.

GoldSim provides several ways to do this. Within the Resources Manager (accessed via **Model|Resources…** or by pressing **F8**), a **Locations** button is provided:



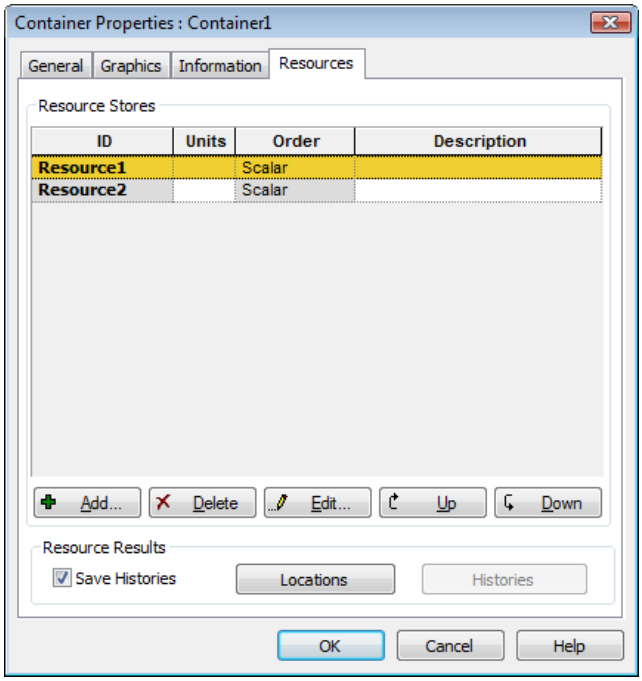
This button lists all of the Resource Stores in the model, along with all of the elements that interact with them:



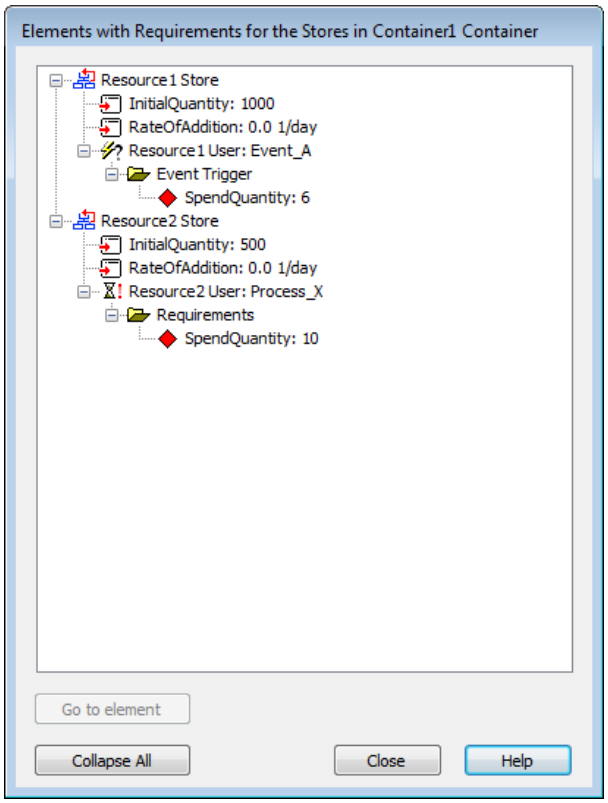
In this example, the model has 3 Resources defined. Resource1 has a Global and a Local Store; Resource2 has Local Store only; Resource3 has a Global Store only. Resource3 does not have any elements that interact with it.

Selecting an element in the tree that uses a Resource and pressing **Go to element** jumps to that element. The **Collapse All** button collapses the tree to the top level (the Resources). (The button then becomes Expand All). Pressing **Ctrl-A** also toggles between expanding and collapsing the tree.

The **Resources** tab for a Container that has Local Stores also has a **Locations** button:



This button lists all of the Resource Stores *in the Container*, along with all of the elements that interact with them:

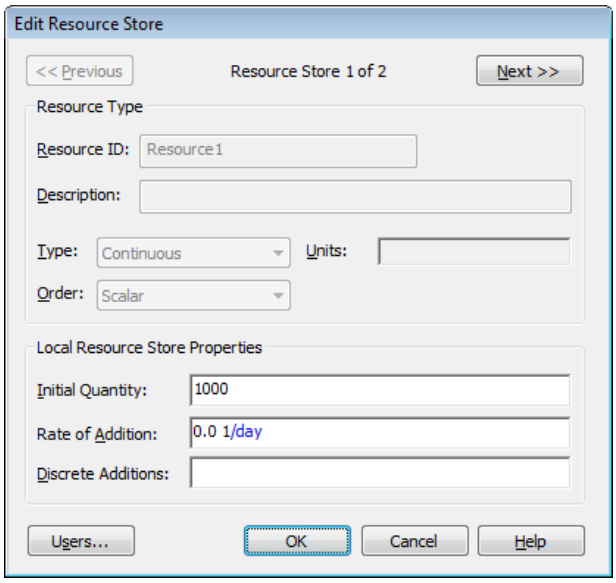


In this example, the Container has 2 Resource Stores defined. Both have a single element that interacts with the Store.

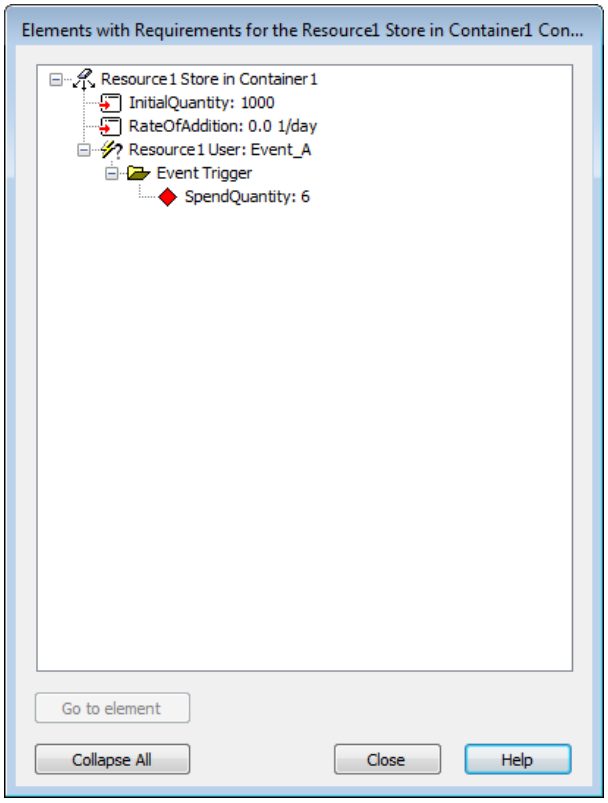
Selecting an element in the tree that uses a Resource and pressing **Go to element** jumps to that element. The **Collapse All** button collapses the tree to

the top level (the Resource Stores). (The button then becomes Expand All). Pressing **Ctrl-A** also toggles between expanding and collapsing the tree.

In the dialog for defining a Resource Store (either globally or locally), there is a **Users** button:



This button lists all of the users (elements) that interact *with that particular Resource Store*:

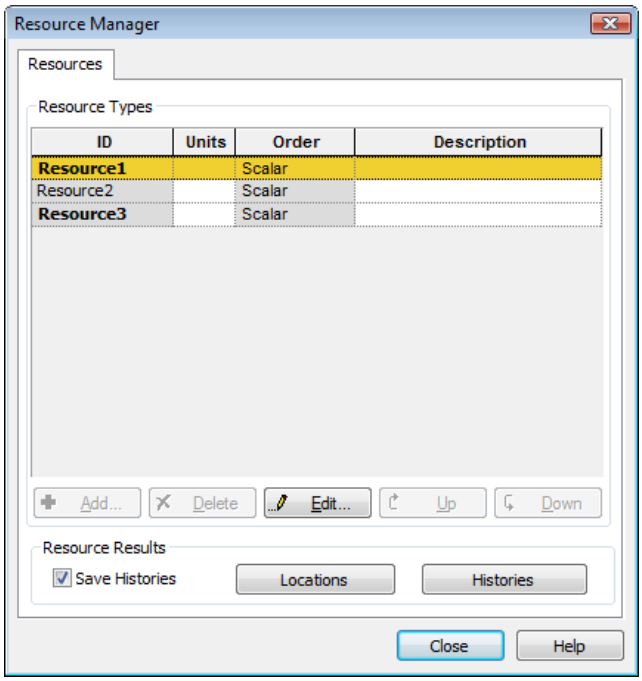


In this example, this Resource Store (Resource1) has one element that interacts with it (Event_A).

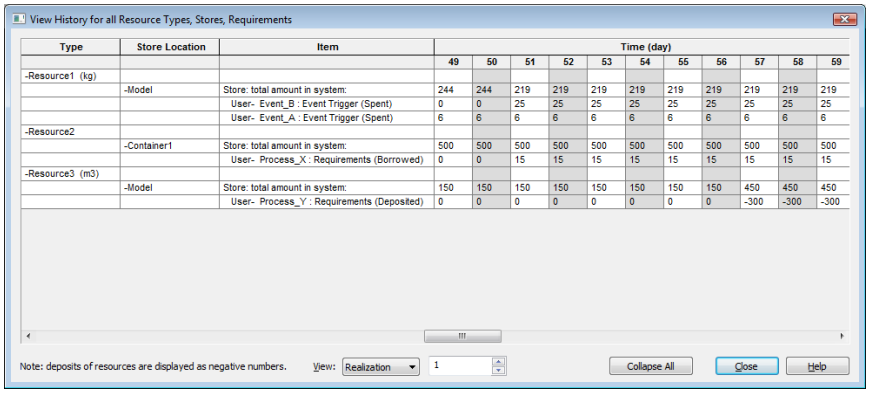
Viewing Resource Results

After you have run a model that uses Resources, you will often want to view how those Resources are being used as a function of time.

GoldSim provides several ways to do this. Within the Resources Manager (accessed via **Model | Resources…** or by pressing **F8**), or from the **Resources** tab of a Container if a Local Store has been defined, a **Histories** button is available when the model is in Result Mode:



This button provides a history (in tabular form) of all of the Resource Stores in the model, along with all of the elements that interact with them (identifying, for example, how much of each Resource Store has been spent, borrowed or deposited):

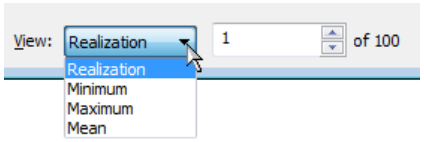


The scroll-bar allows you to scroll horizontally to view all of the time points.

The **Collapse All** button collapses the tree to the top level (the Resources). (The button then becomes **Expand All**). Pressing **Ctrl-A** also toggles between expanding and collapsing the tree.

Note: Deposits (additions) of a Resource appear as negative numbers in this table.

If you have carried out multiple realizations, additional options will be available at the bottom of the Resource Histories dialog. In particular, you must select how you want to view the Monte Carlo results:



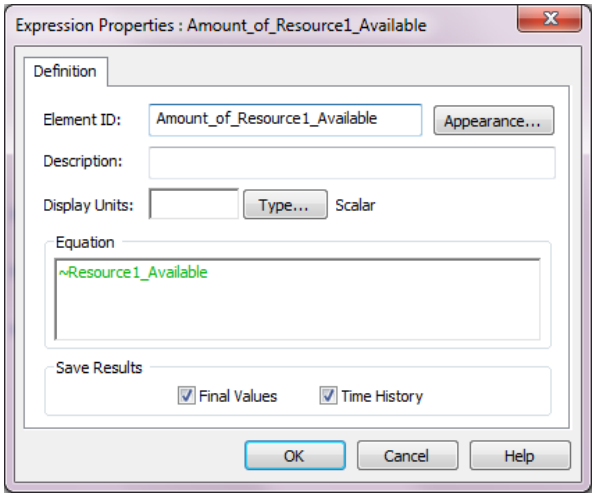
There are four choices:

- You can view a specified realization;
- You can view the minimum value at each time point;
- You can view the maximum value at each time point; or
- You can view the mean value at each time point.

Note: The **Histories** button is grayed out when the model is in Scenario Mode. That is, you cannot view these results when running and comparing scenarios.

Read more: Creating, Running and Comparing Scenarios (page 463).

You can also view time history charts of the amount of each Resource Store available by creating Expression elements that directly reference the locally-available variables for each Resource:



Read more: Referencing Resource Availability and Use in Input Expressions (page 794).

Script Elements

In some situations, you may wish to define a complex function which cannot be readily implemented using the expression editing features supplied by GoldSim. For example, calculation of an output may require very complex logic which would be cumbersome to represent using a Selector element, or it may require a

numerical solution technique (e.g., iteration); or perhaps you need to construct an array using complex logic.

GoldSim provides two separate ways to deal with such situations:

- You can develop separate program modules (written in C, C++, FORTRAN or other compatible programming languages) which can then be directly coupled with the main GoldSim algorithms. These user-defined modules are referred to here as external functions, and the elements through which they are coupled to GoldSim are called External (DLL) elements.
- You can specify a sequence of statements directly within the GoldSim interface using a Script element.

Read more: External (DLL) Elements (page 873).

Although, for some complex calculations, an External (DLL) element may be required, for many applications a Script element can be used. The key advantages of a Script over an External (DLL) is that 1) it does not require use of a separate programming language and interface; and 2) it is much more transparent (all of the "code" can be seen directly in GoldSim).

Scripts are created by inserting and editing statements or statement blocks, which may be variable definition statements, variable assignment statements, statements controlling the sequence of execution in the script (e.g., loops and if statements), or statements used for writing messages to the Run Log. The Script element sequentially evaluates the specified sequence of locally defined statements to determine its output(s).

Note: The Script element does not expect the user to learn or be familiar with a particular language. As a result, scripts are *not* created using a text editor. Rather, statements are inserted and edited within a "controlled environment" within the element's property dialog in which the user selects from a number of available statement types. The syntax is already defined for each type of statement – the user simply specifies the attributes and properties for each statement via a dialog box when the statement is inserted. Statements can subsequently be moved, deleted, and edited.

To provide a quick idea of what a script looks like, a simple example in which a script is used to define a vector is shown below:

Script

	Statement List
1	// Defines items in a vector as a function of the index
2	DO index = 1, 12, 1
3	IF (~Result[~index]<6) THEN
4	Result[~index] = ~index
5	ELSE IF (~Result[~index]<9)
6	Result[~index] = A
7	ELSE
8	Result[~index] = B
9	END IF
10	END DO

The sections below discuss the use of Script elements in detail. The Script element is powerful, and, as a result, somewhat complex. It is strongly recommended that you read through these sections carefully before attempting to build your own scripts.

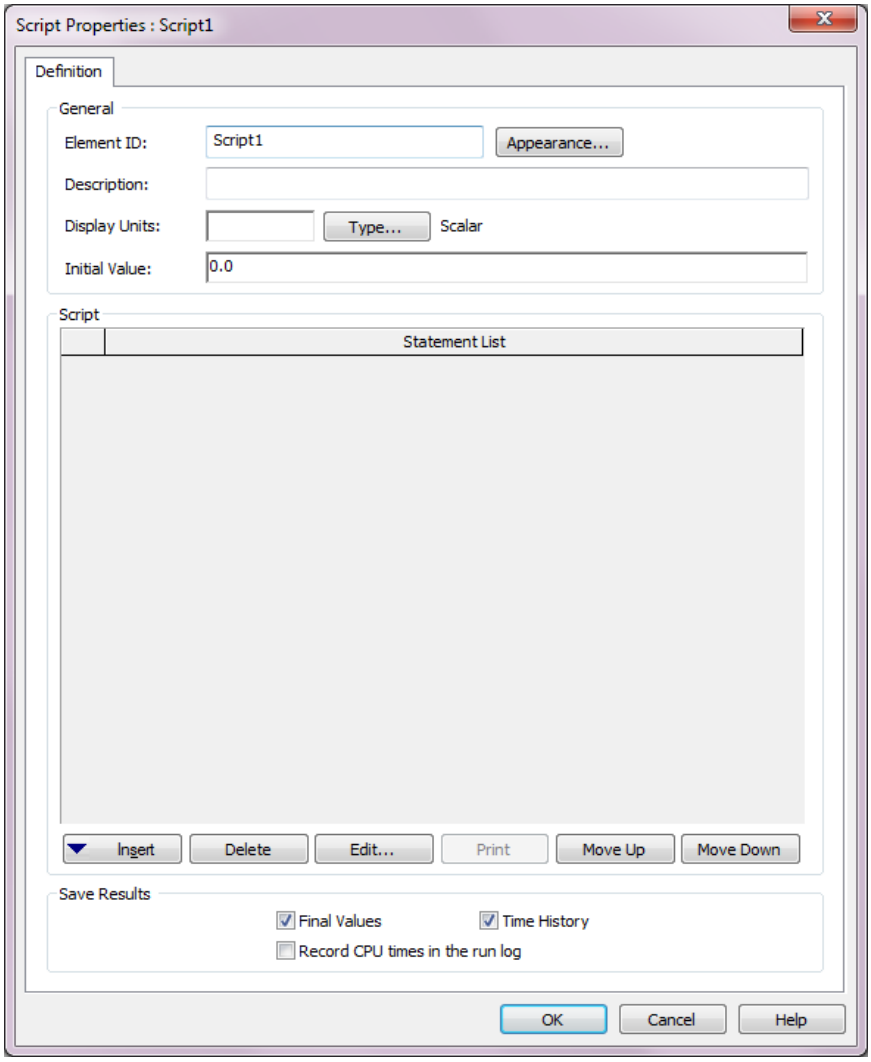
Note: In the sections that follow, the term "Script" (capitalized) refers to the element itself; the term "script" (lower case) refers to the list of statements defined by the user within the element.

Getting Started with the Script Element

The sections below help you get started with the Script element by introducing some basic concepts, and walking through the steps required to build your first script. Before you can learn the advanced Script element features, it is essential that you first understand these basic concepts.

The Script Element Dialog

The properties dialog for a Script element looks like this:



Like all elements, you first specify an **Element ID** and a **Description**.

By default, all Script elements have a single primary output. When defining a Script, therefore, the first step is to define the attributes (units, type, order) of this primary output.

The **Display Units** determine the dimensions of the output. You can specify the type and order by pressing the **Type…** button. The output can be defined as either a value or a condition, and can be specified as a scalar, a vector or a

matrix. By default, the primary output of a new Script element is a scalar, dimensionless value.

Read more: Defining and Assigning Array Variables in a Script (page 812).

Note: It is also possible to define additional outputs for the element (with their own dimensions) when defining local variables within a script.

Read more: Defining Local Script Variables (page 810).

The primary output of a Script element is a ***state variable***. A state variable provides inertia or "memory" to a system because its value is computed based on the historical value of the element's inputs (as opposed to only being a function of the current value of the element's inputs). All state variables have, by definition, an initial value. This allows the output to be computed when there are no historical inputs available (e.g., at the start of simulation).

Read more: Understanding State Variables in GoldSim (page 309); Understanding the Outputs of a Script Element (page 815).

The **Initial Value** input to the Script must have the same attributes (order and dimensions) as the primary output.

Note: The **Initial Value** must be a number or a link from a static variable (e.g., a constant Data element or a Stochastic).

Buttons at the bottom of the dialog are used to insert, delete, move and edit statements in your script. You can also print the entire script.

Read more: Editing Scripts (page 829); Printing Scripts (page 837).

If you check the **Record CPU times in the run log** box, if the element uses more than 1 CPU seconds, a message will be written to the GoldSim run log identifying the element's name, type (i.e., Script), the number of times it was updated, and the total CPU time used.

Read more: The Run Log (page 506).

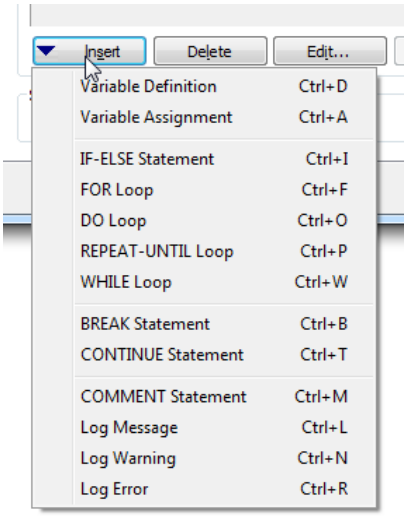
Inserting Statements into a Script

Scripts are created by inserting and editing individual statements (e.g., variable definition statements, variable assignment statements) or statement blocks (e.g., loops, if statements). The Script element sequentially evaluates the specified sequence of locally defined statements to determine its output(s).

It is important to understand that scripts are *not* created using a text editor. Rather, statements are inserted and edited within a "controlled environment" within the element's property dialog in which the user selects from a number of available statement types. The syntax is already defined for each type of statement – the user simply specifies the attributes and properties for each statement via a dialog box when the statement is inserted. Statements can subsequently be moved, deleted, and edited.

There are three ways to insert a statement into a script:

1. Select an existing statement in the script. Press the **Insert** button at the bottom of the Script element dialog. When you press this button, the following choices appear:



Left-click on one of the statement types. It will be inserted *below* the selected statement.

2. Select an existing statement in the script. Press the appropriate hot-key for the statement type (indicated in the dialog above). It will be inserted *below* the selected statement.

   Read more: Hot-keys for Quickly Creating and Editing Scripts (page 830).

3. Right-click in the statement number. This will bring up a context menu, whose first item is **Insert**. This expands to display all of the statement types shown above. Left-click on one of the statement types. It will be inserted *below* the selected statement.

Note: If you hold the **Shift** key while carrying out any of these three methods for inserting a statement, the statement will be inserted *above* the selected statement.

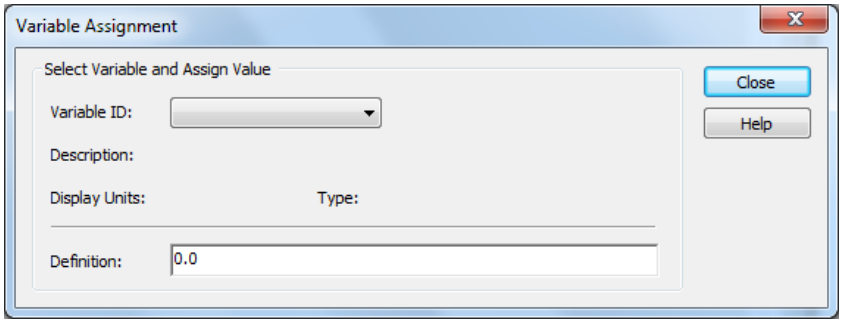
Note: When inserting statement blocks (e.g., If-Else statement, For loops), you can select multiple statements in an existing script (by left-clicking and dragging the line numbers), and GoldSim will "wrap" the statement block around the selected statements. Note, however, that this requires that the selection is a logically complete block of code (e.g., you could not select a portion of another loop).

Read more: Controlling Program Flow in the Script Element (page 816).

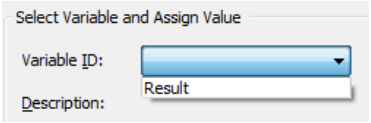
Assigning Values to Script Variables

The most fundamental of the Script element's statements is the Variable Assignment, in which you assign a value (or condition) to a variable (typically in the form of a mathematical expression).

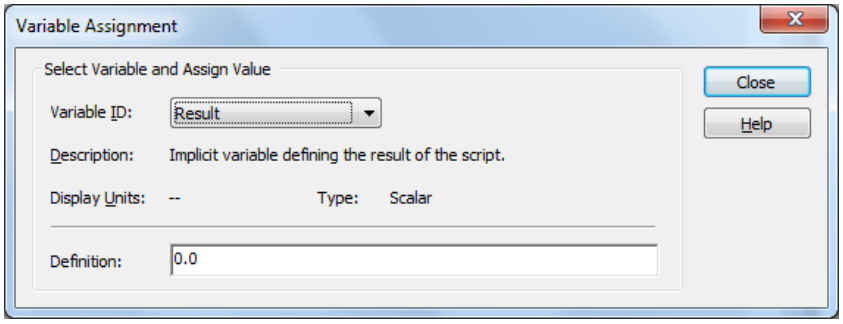
Selecting "Variable Assignment" from the Script element's **Insert** menu (or pressing **Ctrl+A** when in the Script dialog) inserts the statement in the script (below the statement that was selected when this is done), and brings up the following dialog:



It is first necessary to specify which variable is being assigned a value (the **Variable ID**). By default, in a new script, there is only one variable that exists: the primary output. Within a script, this primary output is always referenced as *Result*. Hence, for a new script, this is the only choice in the drop-list:



After selecting the **Variable ID**, GoldSim displays the attributes of the variable (in the example below, the attributes of the primary output of the Script element):



You then specify how the variable is assigned (i.e., in terms of an equation, you are defining the right-hand side of the equation). The **Definition** field is the equivalent of the **Equation** field in an Expression element; you can enter any mathematical expression using all of the functions and operators you can use in other GoldSim input fields.

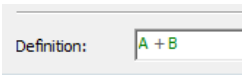
Three different kinds of variables can be referenced in this field:

1. You can reference outputs that are external to the Script element;
2. You can reference the primary output itself (i.e., *Result*); and/or
3. You can reference other local variables that you define in the script.

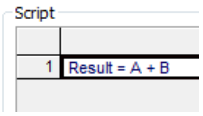
Note: If the Variable you are assigning is an array, the **Definition** portion of the dialog expands to provide some additional options for defining the array.

Read more: Defining and Assigning Array Variables in a Script (page 812).

In the example below, two outputs that are external to the Script element (A and B) are referenced:

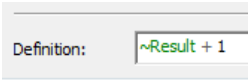


This single statement Script element (which simply adds two numbers) would then look like this:

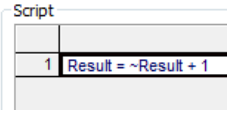


Note: Outside of the Script element, the primary output (*Result*) would be referenced by the Script element's name. It is only referenced as *Result* inside the Script element.

In the example below, the primary output itself is referenced:



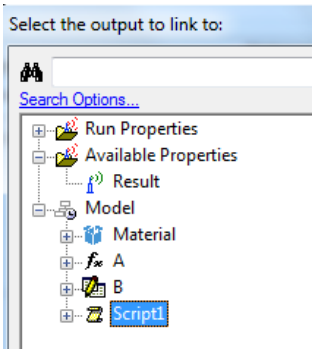
The Script element would then look like this:



In this simple Script element, every time the Script element is updated, it would increment the primary output.

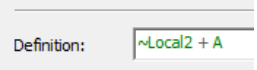
Note that when *Result* is referenced in a statement (i.e., in the right-hand side of the equation), it is referenced using the ~ operator. This is because *Result* is a locally available property. Locally available properties are only available in specific (local) parts of the model. In this case, *Result* can only be referenced inside the Script element, and has no meaning outside the Script element.

When inserting a link while assigning a statement in a script, you will note that *Result* (and any other script variables that you may have defined) are accessible only under Available Properties, and are not listed in the same manner as other outputs:

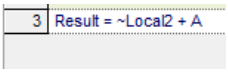


Read more: Understanding Locally Available Properties (page 750).

Finally, in the example below, *Result* is defined in terms of a (previously defined) local script variable (Local2) and an output external to the Script element (A):



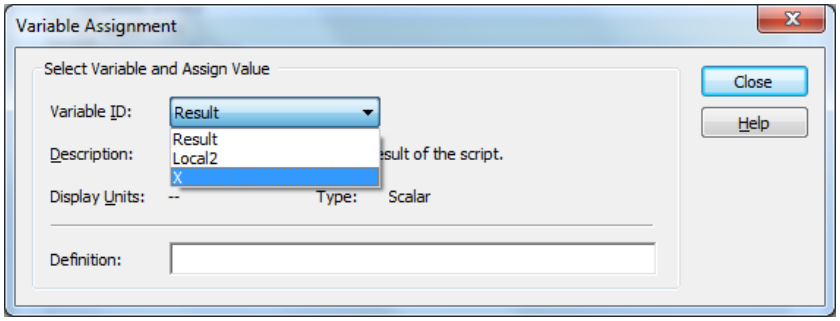
The statement in the Script element would then look like this:



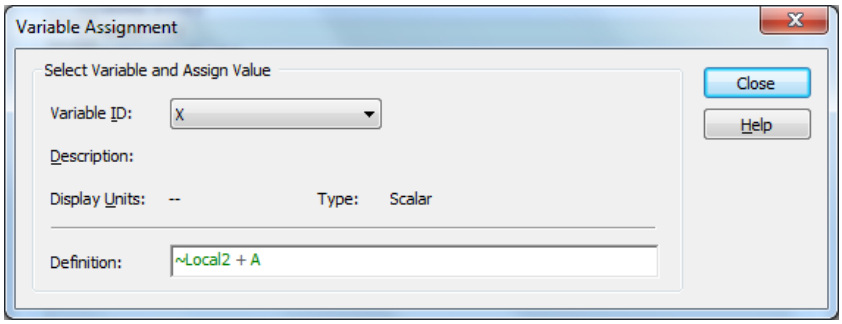
(The next section describes how local variables can be defined and added to a script).

Note that like the *Result* variable, when a local script variable is referenced in a statement, it must be referenced using the ~ operator, since it is a locally available property.

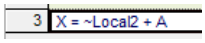
Of course, you can assign values to the local variables in your script (not just the primary output *Result*). For example, in the example below, there are two previously defined local script variables (X and Local2), and the variable X is selected as the **Variable ID** for the statement:



It is then defined in terms of another local script variable (Local2) and an output external to the Script element (A):



The statement in the Script element would then look like this:



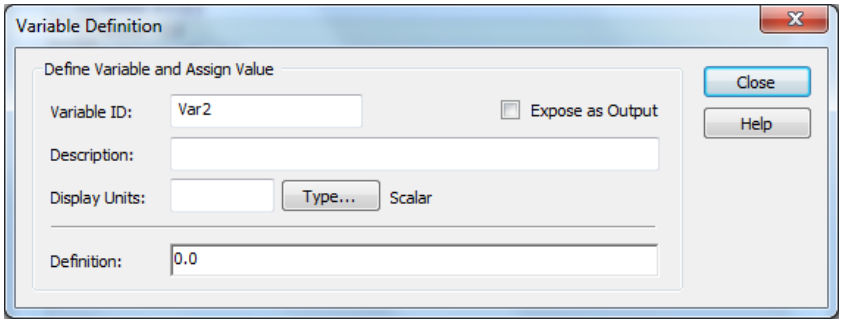
Defining Local Script Variables

Rather than just relying on the primary output (*Result*), most scripts will require local variables to do such things as hold intermediate results or act as counters for loops. Local script variables can also be specified to be additional outputs to the element (so that the Script element has outputs in addition to its primary output).

Script variables are created using a Variable Definition statement. Often a script will start out with multiple Variable Definition statements that create and initialize the local variables that are subsequently used throughout the script.

You add a local variable to a script by selecting "Variable Definition" from the Script element's **Insert** menu, or by simply pressing the hot-key (**Ctrl+D**) from

the main Script dialog. Doing so inserts the statement in the script (below the statement that was selected when this is done), and brings up the following dialog:



You must specify the **ID** of the local variable you are creating, along with its **Display Units** and **Type**. The **Display Units** determine the dimensions of the variable. You can specify the type and order by pressing the **Type…** button. The variable can be a value or a condition, and can be specified as a scalar, a vector or a matrix. By default, a local variable is a scalar, dimensionless value.

Note: If the variable you are defining is an array, the **Definition** portion of the dialog expands to provide some additional options for defining the array.

Read more: Defining and Assigning Array Variables in a Script (page 812).

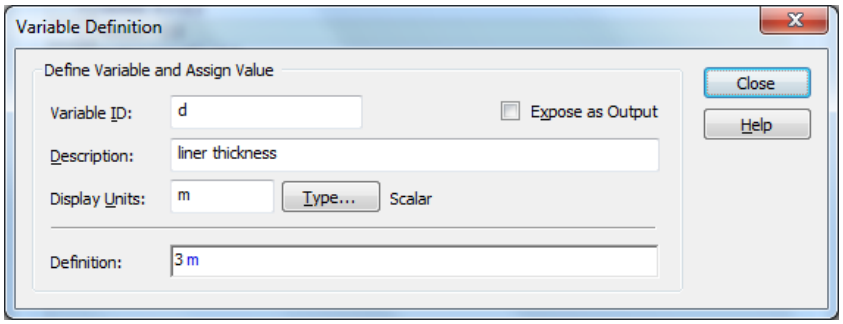
The **Description** is displayed in tool-tips when viewing the script, and is also displayed when modifying the value of the variable using Variable Assignment statements.

The **Definition** field is used to assign a value to the variable. This field is identical to the **Definition** field for a Variable Assignment statement, in that it can reference:

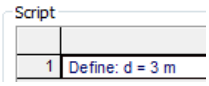
1. Outputs that are external to the Script element;
2. The primary output itself (i.e., Result); and/or
3. Other local variables that you define in the script.

Read more: Assigning Values to Script Variables (page 807).

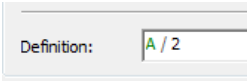
In many cases, however, the assignment in a Variable Definition statement will simply be a constant:



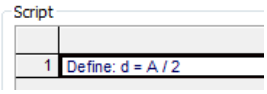
This would then look like this in the script:



Alternatively, you could define the variable and immediately define its value as an expression (as opposed to a constant):



This would then look like this in the script:

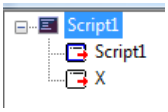


Of course, you can (and almost always will) reassign a definition to a variable later in your script (using a Variable Assignment statement).

Note: User-defined local script variables cannot reference themselves in a Variable Definition statement (e.g., X = ~X + 1 would be invalid, as there is no existing initial value of X to reference in this case). This could, however, be done in a Variable Assignment statement (since X would have been previously defined, and hence could be referenced).

When you create a local variable, you can specify that you want it to be an output for the element by checking the **Expose as Output** checkbox (which by default is cleared). This then adds an output to the Script element.

For example, if a Script element (named Script1) had two local variables (X and Y), and only X was exposed as an output, then outputs of the Script element would look like this:

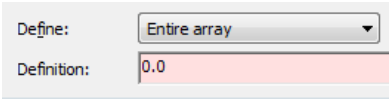


The primary output (referred to as *Result* in the script) would be referenced outside of the Script element as Script1. The secondary output X would be referenced outside of the Script element as Script1.X. The other local script variable (Y) could not be referenced outside of the Script element.

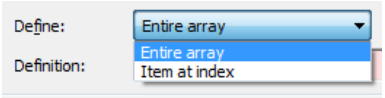
Read more: Understanding the Outputs of a Script Element (page 815).

Defining and Assigning Array Variables in a Script

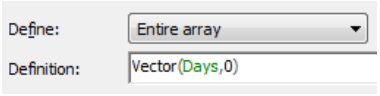
The variables in your script can be defined as arrays (vectors or matrices). When defining a local script variable that is an array in a Variable Definition statement, or assigning a value to a script variable that is an array in a Variable Assignment statement, the **Definition** portion of the dialog expands to provide some additional options for defining the array:



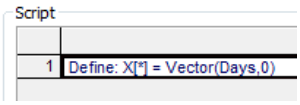
If you are defining a vector, there are two choices in this drop-list:



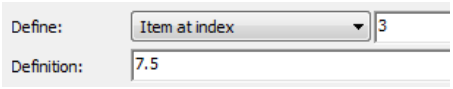
1. If you select the first one ("Entire array"), then the **Definition** field would require the field to be a vector. For example, when defining a variable (named X) that was a vector of *days* you could enter the following to set the entire vector to zero:



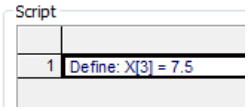
In the script, it would then look like this:



2. Alternatively, if you select "Item at index", then you must also specify the item being specified, and the **Definition** field would require the field to be a scalar. For example, you could enter the following:

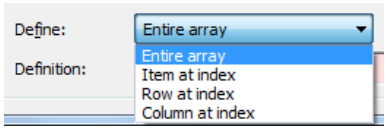


In the script, it would then look like this:

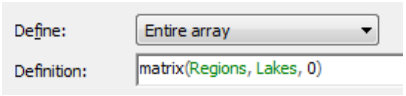


In this case, the third item of the vector would be assigned the value 7.5.

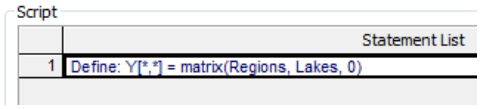
Similarly, if you are defining a matrix, there are four choices in this drop-list:



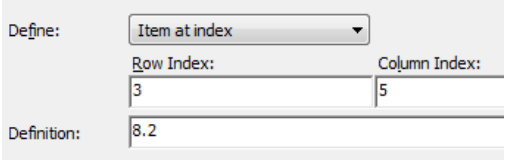
1. If you select the first one ("Entire array"), then the **Definition** field would require the field to be a matrix. For example, when defining a variable (named Y) that was a matrix of *Regions* and *Lakes* you could enter the following to set the entire matrix to 0:



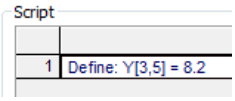
In the script, it would then look like this:



2. Alternatively, if you select "Item at index", then you must also specify the item (row and column) being specified, and the **Definition** field would require the field to be a scalar. For example, you could enter the following:

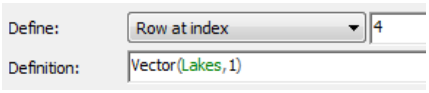


In the script, it would then look like this:

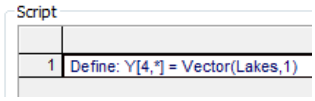


In this case, the item in the 3rd row and 5th column of the matrix would be assigned the value 8.2.

3. If you select "Row at index", then you must also specify the row being specified, and the **Definition** field would require the field to be a vector. For example, you could enter the following:

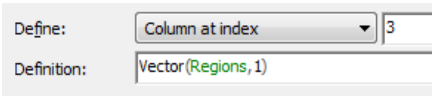


In the script, it would then look like this:

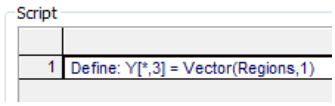


In this case, the item in the 4th row would be assigned as a vector of 1's.

4. If you select "Column at index", then you must also specify the column being specified, and the **Definition** field would require the field to be a vector. For example, you could enter the following:



In the script, it would then look like this:



In this case, the item in the 3rd column would be assigned as a vector of 1's.

Note: When you are specifying the "Item at index" (for a vector or matrix), or "Row at index" or "Column at index" (for a matrix), you do not have to enter a number. You can enter any scalar expression. GoldSim rounds the result to the nearest integer value. This allows you to define arrays by embedding Variable Assignment statements in loops (in which case these indices are often the loop counter variable).

Read more: Controlling Program Flow in the Script Element (page 816); Script Example: Manipulating a Matrix (page 839).

The discussion above applies to assigning values to arrays in both Variable Definition statements and Variable Assignment statements. That is, both types of statements are used to assign values. It is important to understand, however,

that the behavior of the two types of statements is different when assigning values to arrays.

To understand this, consider the following excerpt from a script, containing a Variable Definition statement, as well as a Variable Assignment statement (that references a previously defined variable):

```
2  // This is a Variable Definition statement
3  Define: X[2] = 4
4  // This is a Variable Assignment statement
5  Y[2] = 2
```

Both X and Y are vectors. Line 3 assigns the value 4 to the second item of vector X. Line 5 assigns the value 2 to the second item of vector Y. The question is: since these statements only partially assign values to the vector (i.e., they only assign the second item), how do the statements affect the other items in the vector (that are not explicitly referenced)? The difference is as follows:

- For a *Variable Definition* statement, if it does not assign all the items of the array, the remaining unassigned items are automatically assigned a value of 0 (for value data types) or false (for condition data types).
- For a *Variable Assignment* statement, a partial assignment of a vector or matrix has no effect on the items of the vector or matrix that were not targets of assignment. That is, since all items of the array must have been previously assigned when the variable was defined (and hence already have a value), any items of the array that are not explicitly assigned are unaffected.

Understanding the Outputs of a Script Element

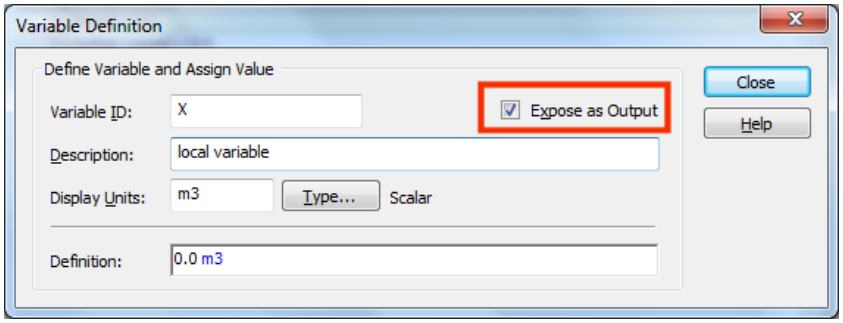
By default, a Script element has a single primary output. The attributes of this output are defined in the main Script dialog.

Read more: The Script Element Dialog (page 805).

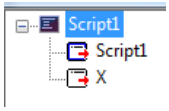
It is also possible to define additional outputs for the element (with attributes and dimensions) when defining local variables within a script.

Read more: Defining Local Script Variables (page 810).

When you create a local variable, you can specify that you want it to be an output for the element by checking the **Expose as Output** checkbox (which by default is cleared):



For example, if a Script element (named Script1) had two local variables (X and Y), and only X was exposed as an output, then outputs of the Script element would look like this:



The primary output (referred to as *Result* in the script) would be referenced outside of the Script element as Script1. The secondary output X would be referenced outside of the Script element as Script1.X. The other local script variable (Y) could not be referenced outside of the Script element.

As is the case for other GoldSim elements, you can check the checkboxes at the bottom of a Script element dialog to save "Final Values" and/or "Time Histories" of all the exposed outputs (including the primary output).

There is an important difference in the behavior of the primary output and any other exposed outputs. The primary output of a Script element is a ***state variable***. A state variable provides inertia or "memory" to a system because its value is computed based on the historical value of the element's inputs (as opposed to only being a function of the current value of the element's inputs). All state variables have, by definition, an initial value. This allows the output to be computed when there are no historical inputs available (e.g., at the start of simulation).

Read more: Understanding State Variables in GoldSim (page 309).

This has an important implication for how the variables can be used in a script. In particular, the value of the primary output is "remembered" between updates of the element. As a result, in a script you could create a Variable Assignment such as "Result = ~Result + 1" as the very first statement. The expression would use the *previous value* of the Result when it was evaluated. This is not possible for any of the other script variables, since they require a Variable Definition statement which initializes the variable every time the Script element is updated.

Note: User-defined local script variables cannot reference themselves in a Variable Definition statement (e.g., X = ~X + 1 would be invalid, as there is no existing initial value of X to reference in this case).

If you would like additional outputs of the Script element to be able to "remember" their previous update (and hence behave like state variables with initial conditions), you can do so by referencing the Script output in a Previous Value element outside of the Script element (and then referencing the Previous Value inside the script).

Read more: Referencing an Output's Previous Value (page 898); Script Example: Referencing a Previous Value Element in a Script (page 840).

Controlling Program Flow in the Script Element

A scripting language is not complete without statements controlling the flow of execution. To this end, the Script element provides a number of statements for controlling programming flow, including if-else statements that support branching of statement execution, and various looping statements to support item-by-item array assignment and iterative calculations.

In particular, the program flow statements provided by the Script element consist of the following:

- If-Else statements;
- For loops;
- Do loops;
- Repeat-Until loops;
- While loops; and

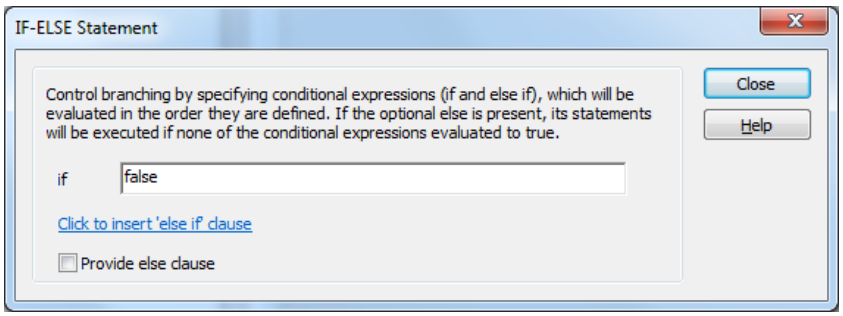
- Break and Continue statements.

These are discussed in the sections below.

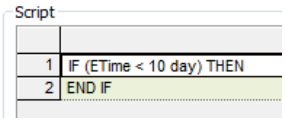
If-Else Statements in Scripts

If-Else statement blocks are used to test conditions that control branching of statement execution within a script. Else-if and Else instructions within the statement block provide a significant amount of control and flexibility.

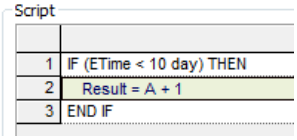
An If-Else statement block can be inserted by selecting "IF-ELSE Statement" from the Script element's **Insert** menu (or pressing **Ctrl+I** when in the Script dialog). The block is inserted below the statement that was selected when this is done. The following dialog will be displayed:



The condition defaults to False. You must enter a statement which evaluates to True or False (e.g., "Etime < 10 day"). After doing so, the script will look like this:



Obviously, such a statement block serves no function unless it contains some statements inside the block. To insert a statement inside the block, you would select the first part of the If-Else block (in this case, line 1), and insert a statement. In the example below, a Variable Assignment statement has been inserted:



In this case, when the Script element was updated, if Etime was less than 10 days, Result would be set to A + 1 (A is an output external to the Script element).

You can also move existing statements that are above or below the If-Else block into the block by using the buttons at the bottom of the Script dialog.

Read more: Moving Statements in Scripts (page 831).

Note: If multiple statements are selected (by left-clicking and dragging the line numbers) when the If-Else statement is first inserted (and the selection is a logically complete block of code), GoldSim will "wrap" the selection in the initial If statement.

In many cases, an If-Else block will have multiple branches. The simplest such case is one in which the original If condition results in one branch if it is True,

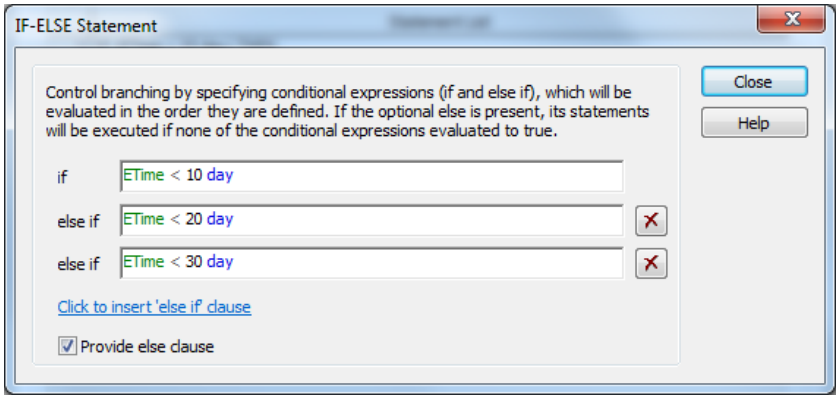
and a separate branch if it is False. This can be implemented by checking **the Provide else clause** box in the If-Else dialog. In the example below, an *else clause* (and a corresponding Variable Assignment) has been added:

Script

```
1  IF (ETime < 10 day) THEN
2    Result = A + 1
3  ELSE
4    Result = A + 2
5  END IF
```

(The Variable Assignment statement could be added to the *else clause* by selecting the ELSE statement (line 3 in this case) and inserting the new statement, or moving it from elsewhere in the script.)

If you need more complex branching, up to four *else if clauses* can be inserted by clicking the **Click to insert "else if" clause** hyperlink on the If-Else dialog. In the example below, two clauses are inserted:



After inserting corresponding Variable Assignments for each clause, the script would look like this:

Script

```
1  IF (ETime < 10 day) THEN
2    Result = A + 1
3  ELSE IF (ETime < 20 day)
4    Result = B
5  ELSE IF (ETime < 30 day)
6    Result = C
7  ELSE
8    Result = A + 2
9  END IF
```

Note: Within an If-Else statement block, *if* and *else if* conditions are evaluated sequentially. As soon as a True statement is found, the statements corresponding to that clause are implemented, and the remaining clauses are skipped. Hence, if two clauses have conditions that evaluate to True, only the statements associated with the first one are implemented.

One key property of If-Else statement blocks that must be understood is that each clause creates local scopes for any variables that are defined within them. For example, consider the following script:

Script

1	IF (ETime < 10 day) THEN
2	Define: Var1 = 0.0
3	Var1 = A + 1
4	ELSE
5	Result = A + 1
6	END IF

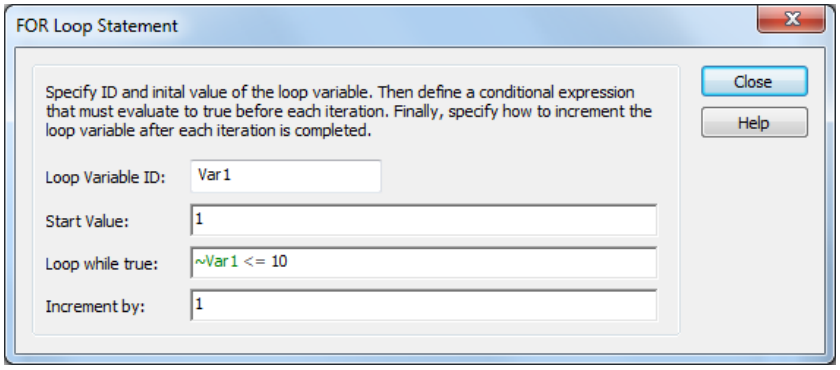
In this example, the local script variable Var1 is created in the first clause. This variable is local to that clause. That is, it cannot be referenced outside of that clause (without redefining it). If you wanted to reference it throughout the entire If-Else block, you would need to define it prior to the If-Else block.

Read more: Understanding Variable Scope in a Script (page 828).

For Loops in Scripts

For loops are one of four different types of loops provided in the Script element. They are used to loop through a set of statements. Common applications are defining and manipulating arrays, and doing an iterative calculation. For loops will be familiar to C/C++ programmers (although you certainly do not need to be familiar with these languages to use them). They require an initial integer value for the loop variable, a "Loop while true" condition, and an integer defining how the loop variable is incremented.

A For loop can be inserted by selecting "FOR Loop" from the Script element's **Insert** menu (or pressing **Ctrl+F** when in the Script dialog). The block is inserted below the statement that was selected when this is done. The following dialog will be displayed:



The first field is the name of the loop variable. This becomes a local variable within the loop (identical to having used a Variable Definition statement to create a variable).

Note: This cannot be the name of an existing script variable. You must define a new variable here.

The remaining three fields are expression fields (i.e., they can be defined as an expression):

Start Value. This is the initial value of the loop variable. It must be specified as a dimensionless scalar value. It is rounded to the nearest integer value.

Loop while true. This must be specified as a scalar condition. This condition is evaluated at the beginning of each loop (including the first loop). If it is True, the loop is executed. If it is false, control drops out the bottom of the loop.

Increment by. After a loop is completed, if the "Loop while true" condition is True, the loop variable is incremented by this value before returning to the top of the loop. It must be specified as a dimensionless scalar value (it can be positive or negative). It is rounded to the nearest integer value.

Note: If you do not change the default settings, the loop will be carried out 10 times.

Note: When the loop variable is referenced (either in the loop definition or any statements that are subsequently added to the contents of the loop), it is referenced using the ~ operator. This is because the loop variable, like all local script variables, is a locally available property. Locally available properties are only available in specific (local) parts of the model.

Read more: Understanding Locally Available Properties (page 750).

After defining the properties of the For loop, the script will look like this:

Script

	Statement List
1	FOR (Var1 = 1; ~Var1 <= 10; Var1 = ~Var1 + 1)
2	END FOR

Note that the first line indicates the start value, loop while true condition, and the increment.

Obviously, such a loop serves no function unless it contains some statements inside the looping block. To insert a statement inside the block, you would select the first part of the block (in this case, line 1), and insert a statement. In the example below, a Variable Assignment statement has been inserted:

Script

	Statement List
1	FOR (Var1 = 1; ~Var1 <= 10; Var1 = ~Var1 + 1)
2	Result = ~Result + 1
3	END FOR

In this case, every time the Script element was updated, Result would be incremented by 1.

You can also move existing statements that are above or below the loop into the loop by using the buttons at the bottom of the Script dialog.

Read more: Moving Statements in Scripts (page 831).

Note: If one or more statements are selected (by left-clicking and dragging the line numbers) when the For loop is first inserted (and the selection is a logically complete block of code), GoldSim will "wrap" the selection in loop.

Because the loop variable is simply a local script variable, it can also be assigned values within the loop (i.e., in addition to being incremented at the end of each loop). For example, this loop would be interrupted if A = B:

Script

	Statement List
1	FOR (Var1 = 1; ~Var1 <= 10; Var1 = ~Var1 + 1)
2	Result = ~Result + 1
3	Var1 = if(A=B, 11,~Var1)
4	END FOR

For loops can have any number of statements within them. For example, a For loop could contain other nested loops, as well as complex if, then logic. Here is an example of a For loop with a nested For loop and an If-Else block:

Script

```
1  FOR (X = 1; ~X <= 100; X = ~X + 1)
2    FOR (Y = 1; ~Y <= 20; Y = ~Y + 1)
3      IF (~X < 50) THEN
4        Result[~X,~Y] = ~Y
5      ELSE
6        Result[~X,~Y] = 2*~Y
7      END IF
8    END FOR
9  END FOR
```

Read more: If-Else Statements in Scripts (page 817).

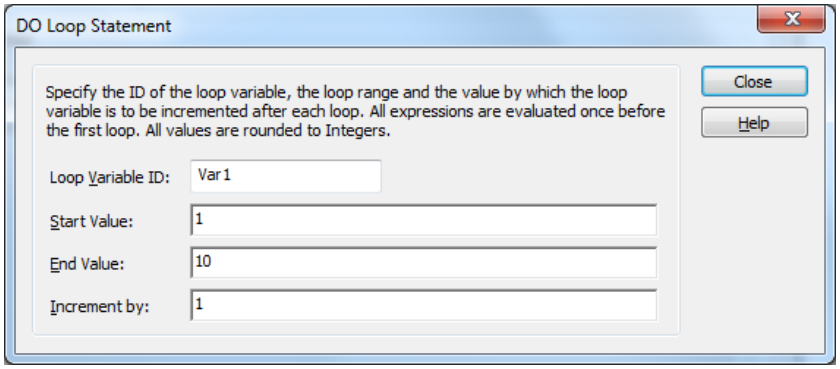
One key property of For loops that must be understood is that the loop creates a local scope for any variables that are defined within it. Hence, the loop variable (and any other variables defined in the loop) could not be referenced outside of the loop (without first redefining them).

Read more: Understanding Variable Scope in a Script (page 828).

Do Loops in Scripts

Do loops are one of four different types of loops provided in the Script element. They are used to loop through a set of statements. Common applications are defining and manipulating arrays, and doing an iterative calculation. Do loops will be familiar to FORTRAN programmers (although you do not need to be familiar with that language to use them). They require an initial integer value for the loop variable, an end value for the variable and an integer defining how the loop variable is incremented.

A Do loop can be inserted by selecting "DO Loop" from the Script element's **Insert** menu (or pressing **Ctrl+O** when in the Script dialog). The block is inserted below the statement that was selected when this is done. The following dialog will be displayed:



The first field is the name of the loop variable. This becomes a local variable within the loop (identical to having used a Variable Definition statement to create a variable).

Note: This cannot be the name of an existing script variable. You must define a new variable here.

The remaining three fields are expression fields (i.e., they can be defined as an expression):

Start Value. This is the initial value of the loop variable. It must be specified as a dimensionless scalar value. It is rounded to the nearest integer value.

End Value. This is the end value of the loop variable. It must be specified as a dimensionless scalar value. It is rounded to the nearest integer value.

Increment by. This is the increment of the loop variable. It must be specified as a dimensionless scalar value. It is rounded to the nearest integer value.

At the beginning of the first loop, the total number of loops to be carried out is computed as follows:

(**End Value** - **Start Value** + **Increment by**)/ **Increment by**

Three points should be noted here:

- If the number of loops computed above is negative, no loops are carried out;
- If **Increment by** is positive and **Start Value** > **End Value**, no loops are carried out; and
- If **Increment by** is negative and **Start Value** < **End Value**, no loops are carried out.

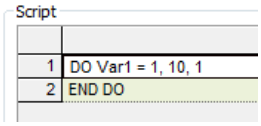
Note: Because the number of loops is computed at the beginning of the first loop, changing the Loop Variable in the middle of the loop has no impact on the number of loops carried out.

Note: If you do not change the default settings, the loop will be carried out 10 times.

Note: When the loop variable is referenced (i.e., in any statements that are subsequently added to the contents of the loop), it is referenced using the ~ operator. This is because the loop variable, like all local script variables, is a locally available property. Locally available properties are only available in "locally available" parts of the model.

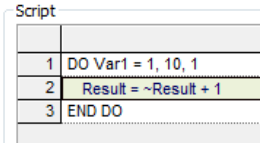
Read more: Understanding Locally Available Properties (page 750).

After defining the properties of the Do loop, the script will look like this:



Note that the first line indicates the start value, the end value, and the increment.

Obviously, such a loop serves no function unless it contains some statements inside the looping block. To insert a statement inside the block, you would select the first part of the block (in this case, line 1), and insert a statement. In the example below, a Variable Assignment statement has been inserted:



In this case, every time the Script element was updated, Result would be incremented by 10. Hence, if the model had 10 timesteps, at the end of the simulation, Result would equal 110, since the loop would be evaluated 11 times (the loop is evaluated at Etime = 0 and every subsequent timestep).

You can also move existing statements that are above or below the loop into the loop by using the buttons at the bottom of the Script dialog.

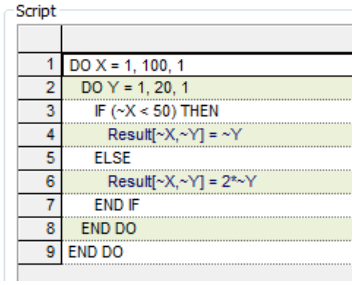
Read more: Moving Statements in Scripts (page 831).

Note: If multiple statements are selected (by left-clicking and dragging the line numbers) when the Do loop is first inserted (and the selection is a logically complete block of code), GoldSim will "wrap" the selection in the loop.

Warning: Unlike For loops, in a Do loop, the loop variable should generally *not* be assigned values within the loop. If it is, it will have no effect on the number of loops since the original increment logic is restored at the bottom of each loop.

Read more: For Loops in Scripts (page 819).

Do loops can have any number of statements within them. For example, a Do loop could contain other nested loops, as well as complex if, then logic. Here is an example of a Do loop with a nested Do loop and an If-Else block:



Read more: If-Else Statements in Scripts (page 817).

One key property of Do loops that must be understood is that the loop creates a local scope for any variables that are defined within it. Hence, the loop variable (and any other variables defined in the loop) could not be referenced outside of the loop (without first redefining them).

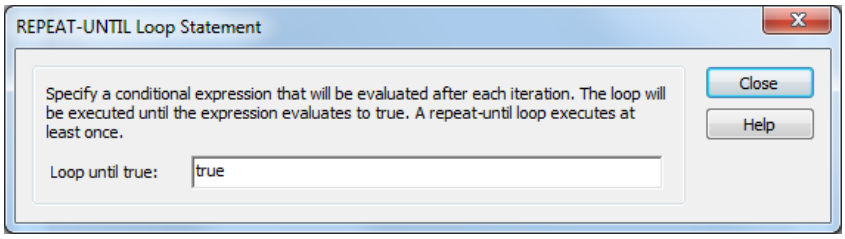
Read more: Understanding Variable Scope in a Script (page 828).

Repeat-Until Loops in Scripts

Repeat-Until loops are one of four different types of loops provided in the Script element. They are used to loop through a set of statements. Common

applications are defining and manipulating arrays, and doing an iterative calculation. Repeat-Until loops require a single input: a condition that specifies whether the loop is to continue (which is evaluated at the bottom of the loop). As a result, a Repeat-Until loop is always evaluated at least once.

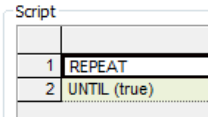
A Repeat-Until loop can be inserted by selecting "REPEAT-UNTIL Loop" from the Script element's **Insert** menu (or pressing **Ctrl+P** when in the Script dialog). The block is inserted below the statement that was selected when this is done. The following dialog will be displayed:



Loop until true must be specified as a scalar condition. This condition is evaluated at the end of each loop. If it is False, another loop is carried out. If it is True, control drops out the bottom of the loop.

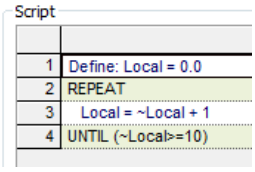
Note: If the condition is initially True (the default), one loop will be carried out.

After defining the properties of the Repeat-Until loop, the script will look like this:



Note that the last line indicates the "loop until" condition.

Obviously, such a loop serves no function unless it contains some statements inside the looping block. To insert a statement inside the block, you would select the first part of the block (in this case, line 1), and insert a statement. In the example below, a Variable Assignment statement has been inserted (and a condition has been defined):



In general, the condition for the Repeat-Until loop will involve a local variable that is also being assigned inside the loop. In this example, when control exits the loop, the variable Local would be incremented by 10.

You can also move existing statements that are above or below the loop into the loop by using the buttons at the bottom of the Script dialog.

Read more: Moving Statements in Scripts (page 831).

Note: If multiple statements are selected (by left-clicking and dragging the line numbers) when the Repeat-Until loop is first inserted (and the selection is a logically complete block of code), GoldSim will "wrap" the selection in the loop.

Repeat-Until loops can have any number of statements within them. For example, a Repeat-Until loop could contain other nested loops, as well as complex if, then logic. Here is an example of a Repeat-Until loop with a nested Repeat-Until loop and an If-Else block:

Script

```
1  Define: X = 1
2  Define: Y = 1
3  REPEAT
4    REPEAT
5      IF (~X < 50) THEN
6        Result[~X,~Y] = ~Y
7      ELSE
8        Result[~X,~Y] = 2*~Y
9      END IF
10     Y = ~Y+1
11   UNTIL (~Y >= 20)
12   X = ~X + 1
13 UNTIL (~X >= 100)
```

Read more: If-Else Statements in Scripts (page 817).

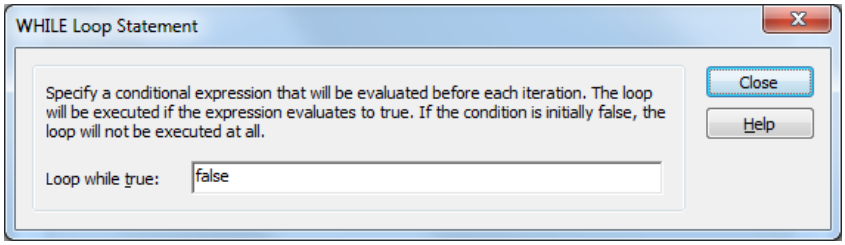
One key property of Repeat-Until loops that must be understood is that the loop creates a local scope for any variables that are defined within it. Hence any variables defined in the loop could not be referenced outside of the loop (without first redefining them).

Read more: Understanding Variable Scope in a Script (page 828).

While Loops in Scripts

While loops are one of four different types of loops provided in the Script element. They are used to loop through a set of statements. Common applications are defining and manipulating arrays, and doing an iterative calculation. While loops are a common construct in a number of programming languages. They require a single input: a condition that specifies whether the loop is to continue (the condition is evaluated at the top of the loop).

A While loop can be inserted by selecting "WHILE Loop" from the Script element's **Insert** menu (or pressing **Ctrl+W** when in the Script dialog). The block is inserted below the statement that was selected when this is done. The following dialog will be displayed:



Loop while true must be specified as a scalar condition. This condition is evaluated at the beginning of each loop. If it is True, another loop is carried out. If it is False, control drops out the bottom of the loop.

Note: If the condition is initially False (the default), no loops will be carried out at all.

After defining the properties of the While loop, the script will look like this:

Script

1	WHILE (false)
2	END WHILE

Note that the first line indicates the loop's "while" condition.

Obviously, such a loop serves no function unless it contains some statements inside the looping block. To insert a statement inside the block, you would select the first part of the block (in this case, line 1), and insert a statement. In the example below, a Variable Assignment statement has been inserted (and a condition has been defined):

Script

1	Define: Local = 0.0
2	WHILE (~Local<10)
3	Local = ~Local+1
4	END WHILE

In general, the condition for the While loop will involve a local variable that is also being assigned inside the loop. In this example, when control exits the loop, the variable Local would be incremented by 10.

You can also move existing statements that are above or below the loop into the loop by using the buttons at the bottom of the Script dialog.

Read more: Moving Statements in Scripts (page 831).

Note: If multiple statements are selected (by left-clicking and dragging the line numbers) when the While loop is first inserted (and the selection is a logically complete block of code), GoldSim will "wrap" the selection in the loop.

While loops can have any number of statements within them. For example, a While loop could contain other nested loops, as well as complex if, then logic. Here is an example of a While loop with a nested While loop and an If-Else block:

Script

1	Define: X = 1
2	Define: Y = 1
3	WHILE (~X<=100)
4	WHILE (~Y<=20)
5	IF (~X < 50) THEN
6	Result[~X,~Y] = ~Y
7	ELSE
8	Result[~X,~Y] = 2*~Y
9	END IF
10	Y = ~Y+1
11	END WHILE
12	X = ~X + 1
13	END WHILE

Read more: If-Else Statements in Scripts (page 817).

One key property of While loops that must be understood is that the loop creates a local scope for any variables that are defined within it. Hence any variables defined in the loop can not be referenced outside of the loop (without first redefining them).

Read more: Understanding Variable Scope in a Script (page 828).

Break and Continue Statements in Scripts

Break and Continue statements are used within loops (For, Do, While and Repeat-Until) to redirect script execution. These statements can only be used inside a loop. Outside of a loop, they have no meaning and will generate an error.

A Break statement can be inserted by selecting "BREAK Statement" from the Script element's **Insert** menu (or pressing **Ctrl+B** when in the Script dialog). A Continue statement can be inserted by selecting "CONTINUE Statement" from the Script element's **Insert** menu (or pressing **Ctrl+T** when in the Script dialog). There are no inputs for a Break or Continue statement; hence there is no dialog. When you insert one of these statements, they are simply added to the script.

Break statements break out of the loop. That is, they redirect execution to the statement *following* the loop in which they are contained. Hence, Break statements are usually placed within an if statement inside of a loop to conditionally exit the loop earlier than planned.

For example, in the script below, when ETime is equal to 0 (i.e., the first time the element is updated), the loop will be broken on the second time through (i.e., when n = 2). As a result, *Result* will only be incremented once.

Script

```
1  DO n = 1, 10, 1
2    IF (ETime = 0 day and ~n > 1) THEN
3      BREAK
4    END IF
5    Result = ~Result+1
6  END DO
```

Note: A Break statement within nested loops breaks the nearest enclosing loop

Continue statements transfer control to the bottom of the enclosing loop. That is, they do not break out of the loop completely, they simply skip the statements between the Continue statement and the bottom of the loop.

For example, in the script below, a vector (X) is being defined in a Do loop. In this case, the first 5 items of the vector are not assigned (they are skipped). Items 6 through 20, however, are assigned.

Script

```
1  Define: X[*] = vector(0)
2  DO n = 1, 20, 1
3    IF (~n <= 5) THEN
4      CONTINUE
5    END IF
6    X[~n] = ~n
7  END DO
```

Note: A Continue statement within nested loops transfers control to the bottom of the nearest enclosing loop.

Understanding Variable Scope in a Script

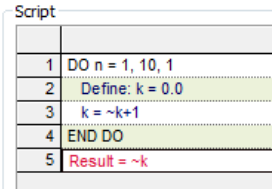
In order to effectively use Script elements in GoldSim, it is important to understand the concept of **variable scope**.

When a variable is defined in a script (either by a Variable Definition statement, or by a Do or For loop, which implicitly define loop variables), the variable persists (and hence can be referenced) from the time it is defined to the statement that closes the scope in which the variable is defined.

The scope of script variables that are 1) not inside a loop; and 2) not inside an If-Else statement consists of all lines of the script below the definition. That is, they persist until the last statement in the script.

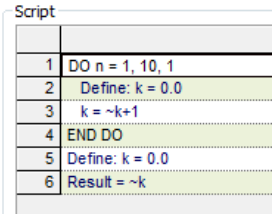
However, script variables defined in loops or If-Else statements do not persist throughout the remainder of the script. Their scope is limited to the the loop or If-Else clause where they are defined.

To illustrate this, let's first consider a simple Do loop:

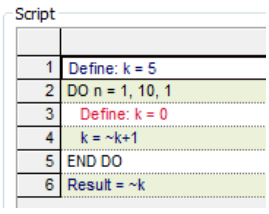


Within this script, there are actually two variables that have been defined: 1) n is defined implicitly by the Do loop; and 2) k is defined explicitly by a Variable Definition statement. For both of these variables, the scope is the Do loop itself. The scope does not extend beyond the Do loop. Hence, neither n nor k can be referenced outside the loop. In the example, k is referenced outside of the loop, and this generates an error (line 5 is displayed in red, and the model will not run).

In order to reference k outside the loop, it would have to be redefined *after* the loop:



In the example above, k ceased to exist after line 4, and then was redefined at line 5. Hence, the two scopes do not overlap. Note that defining k before (and within the loop) would have generated an error. That is, the following script is invalid:



In this case, line 3 is displayed in red and marked as invalid. The reason for this error is that since k was defined in line 1, its scope is the entire script. Hence, when we try to define it again in line 3, an error is generated because the two scopes overlap. Defining the same variable twice within the same scope

generates a redefinition error. In the previous example, there was no error because the scopes did not overlap.

This same logic also applies for the other looping constructs in GoldSim (For loops, While loops, and Repeat-Until loops).

Read more: Controlling Program Flow in the Script Element (page 816).

If-Else statement blocks also create local scopes. In this case, the scope of any variables inside an If-Else statement block is not the entire block; rather, each clause creates local scopes for any variables that are defined within them. For example, consider the following script:

Script

1	IF (ETime < 10 day) THEN
2	Define: Var1 = 0.0
3	Var1 = A + 1
4	ELSE
5	<Var1>[?] = A + 2
6	END IF

In this example, the local script variable Var1 is created in the first clause. The scope of this variable is only that clause (lines 2 and 3). Hence, it cannot be referenced outside of that clause (without redefining it). Referencing it in the second clause generates an error.

If you wanted to reference it throughout the entire If-Else block, you would need to define it prior to the If-Else block:

Script

1	Define: Var1 = 0.0
2	IF (ETime < 10 day) THEN
3	Var1 = A + 1
4	ELSE
5	Var1 = A + 2
6	END IF

Editing Scripts

Scripts are created by inserting and editing individual statements (e.g., variable definition statements, variable assignment statements) or statement blocks (e.g., loops, if statements).

Statements are inserted and edited within a "controlled environment" within the element's property dialog in which the user selects from a number of available statement types. The syntax is already defined for each type of statement – the user simply specifies the attributes and properties for each statement via a dialog box when the statement is inserted. Statements can subsequently be moved, deleted, and edited.

Read more: Inserting Statements into a Script (page 806).

The active line of the script is outlined in black.

Script

1	Define: Var1 = 0.0
2	IF (ETime < 10 day) THEN
3	Var1 = A + 1
4	ELSE
5	Var1 = A + 2
6	END IF

Line 3 (outlined in black) is the active line.

To change the active line of the script, you can use the Up and Down arrow keys, or click on a statement of the script using the mouse. Holding the **Ctrl** key down while pressing the Up or Down arrow jumps to the top or bottom of the script, respectively. Once a statement is highlighted, it can be opened for editing by double-clicking or pressing **Alt-Enter**.

A valid script statement is shown in black text. Invalid statements are shown in red text.

The sections below discuss hot-keys provided to quickly insert statements and edit scripts, as well as how to delete and move script statements.

Hot-keys for Quickly Creating and Editing Scripts

GoldSim provides a number of hot-keys for accessing common Script element functions directly from the keyboard:

Hot-key	Description
Ctrl+D	Inserts a Variable Definition statement
Ctrl+A	Inserts a Variable Assignment statement
Ctrl+I	Inserts If-Else statement block
Ctrl+F	Inserts FOR loop statement block
Ctrl+O	Inserts DO loop statement block
Ctrl+P	Inserts REPEAT-UNTIL loop statement block
Ctrl+W	Inserts WHILE loop statement block
Ctrl+M	Inserts a Comments statement
Ctrl+L	Inserts a Log Message statement
Ctrl+N	Inserts a Log Warning statement
Ctrl+R	Inserts a Log Error statement
Ctrl+Delete	Deletes selected statement(s)
Ctrl+Shift+Delete	Deletes all statements
Cursor Up	Select previous statement
Cursor Down	Select next statement
Ctrl+Cursor Up	Selects first statement
Ctrl+Cursor Down	Selects last statement
Alt+Enter	Edits active statement (opens dialog)
Ctrl+Shift+Cursor Up	Moves active statement up by one row
Ctrl+Shift+Cursor Down	Moves active statement down by one row

In addition to these, several additional hot-keys are used for debugging scripts.

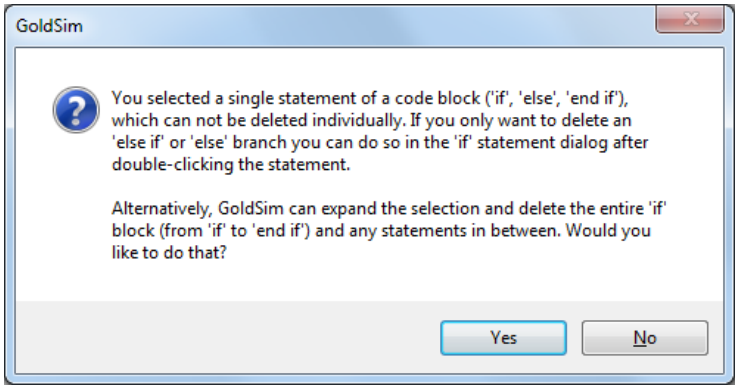
Read more: Debugging Scripts (page 833).

Note: If you use one of the statement insertion hot-keys, the statement(s) will be inserted immediately *after* the active line of the script. If you use one of the insertion hot-key combinations *while holding down the shift key*, the inserted statement(s) will appear before rather than after the active line.

Note: When inserting statement blocks (e.g., If-Else statement, For loops), you can select one or more statements in an existing script (by left-clicking and dragging the line numbers), and when you use insertion hot-keys, GoldSim will "wrap" the statement block around the selected statements. Note, however, that this requires that the selection is a logically complete block of code (e.g., you could not select a portion of another loop).

Deleting Statements from Scripts

If a single line of a script is highlighted, it can be deleted by pressing **Ctrl+Delete** or using the **Delete** button at the bottom of the Script dialog. If you try to delete a single line that is part of a loop definition or an If-Else statement block, however, you will see a message similar to this:



As can be seen, the dialog asks if you want GoldSim to automatically expand the selection to delete the entire loop or if statement (Yes), or cancel the deletion (No).

If you would like to delete a contiguous block of statements, you can select them by highlighting their line numbers (by clicking and dragging). Once you have selected them this way, you can press **Ctrl+Delete** or use the **Delete** button on the Script element. Note, however, when you are deleting multiple statements, the highlighted block of statements for deletion must be a logically complete block of statements. For example, you could not delete just a portion of a For loop or If-Else statement block.

Note: Ctrl-Shift-Delete will delete all statements in a script.

Moving Statements in Scripts

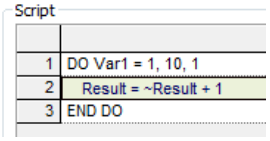
One or more statements can be moved by highlighting them and pressing **Ctrl+Shift+Up** or **Ctrl+Shift+Down**, or by using the **Move Up** or **Move Down** buttons at the bottom of the Script dialog.

There are, however, several rules that determine when GoldSim will allow a given selection to be moved. A statement that opens a scope, such as an IF statement or DO loop, cannot be moved to below the statement that closes the scope (e.g., the corresponding ELSE, ELSE IF, of End IF statement for an IF or the END DO for a DO loop). Similarly, a statement that closes a scope cannot be moved up to above the statement that opens a scope.

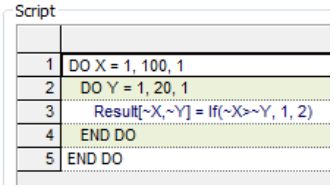
Read more: Understanding Variable Scope in a Script (page 828).

For example, in the script shown below, the line 1 could not be moved below line 3 because doing so would move a scope-opening statement below the

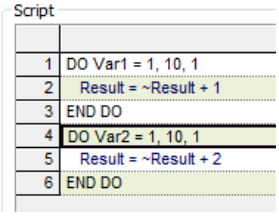
corresponding scope-closing statement. Similarly, line 3 could not be moved above line 1.



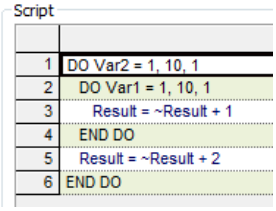
A selection of an incomplete scope, such as one that includes a DO statement without the corresponding END DO statement, cannot be moved into a higher scope or a lower nested scope, as this would create an illogical script where the scope-opening statement becomes separated from its corresponding scope-closing statement. For example, in the script shown below, line 2 cannot be moved up and line 4 cannot be moved down, but if you selected lines 2 through 4 at the same time, you would be able to move the entire Do loop statement block.



If you select an incomplete scope and attempt to move it up/down into another scope that is at the same level, the selection will be moved up/down *past* the other scope, which will have the result that the scopes will be nested. As an example of this, consider the following script:



If you try to move line 4 up, it is moved all the way to the top, resulting in nested Do loops:

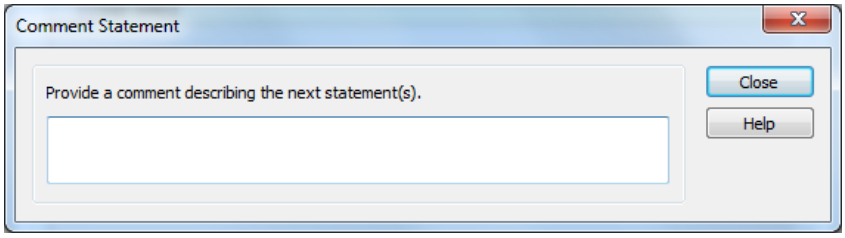


Documenting Scripts

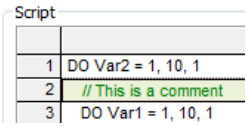
Particularly for complex scripts, it is very important that you document them internally. Comment statements provide an effective way to do this.

Comments can be inserted throughout scripts to document the statements and logic.

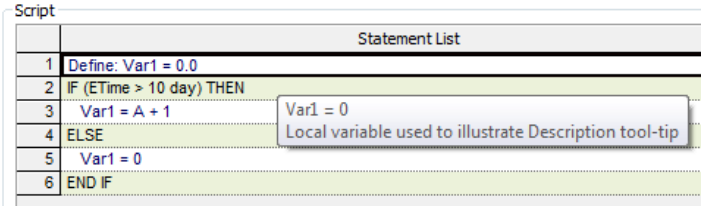
Comments can be inserted by selecting "COMMENT statement" from the Script element's **Insert** menu (or pressing **Ctrl+M** when in the Script dialog). The comment is inserted below the statement that was selected when this is done. The following dialog will be displayed:



Comments can be up to 100 characters long. They appear in the script as green text, prefaced by a double forward slash:



It should also be pointed out that the Description entered when a local variable is created in a script also serves as an excellent documentation tool. These descriptions are shown in tool-tips when you hover over a Variable Definition or Variable Assignment statement:

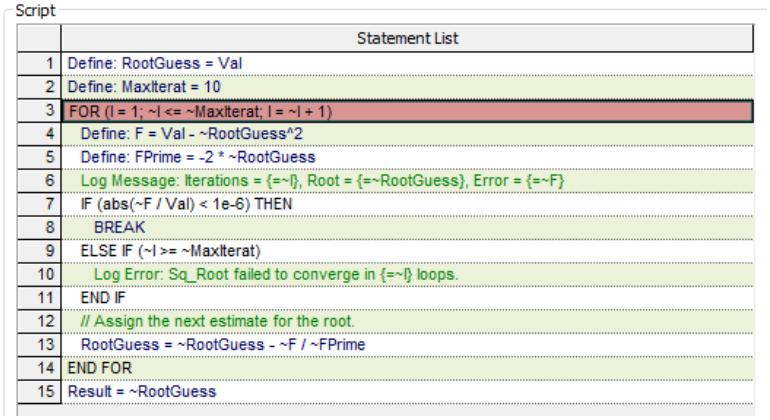


Debugging Scripts

In order to create and test scripts (particularly those with complex branching and looping), it is necessary to be able to step through the script to debug it.

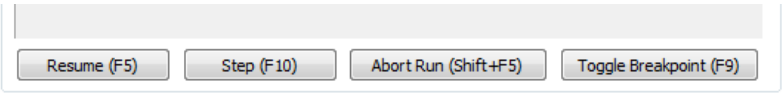
GoldSim facilitates this by allowing you to create ***breakpoints*** in your script. A breakpoint provides a mechanism for pausing a simulation in the middle of a script, and subsequently allowing you to step forward (line by line), abort the calculation, or resume the calculation.

To insert a breakpoint in a script, select (highlight) the line where you would like to insert the breakpoint and press **F9**, or right-click and select **Breakpoint**. (You can remove a breakpoint by pressing **F9**, or right-clicking and selecting **Breakpoint** again). Lines with breakpoints have a red background in the script statement list. In the example below, there is a single breakpoint inserted (at line 3):



When you run the model, at the point when a breakpoint is reached, the model will pause, and the Script element will be opened.

When a script is paused at a breakpoint, you will notice that the buttons at the bottom of the Script dialog differ from those that are displayed when the model is in Edit Mode:



The *next* line to be evaluated is outlined in black. Hence, when you first reach a breakpoint, the line where the breakpoint was inserted will be outlined in black. Note that a line that is outlined in black *has not yet been evaluated*. The calculation is actually paused immediately prior to that line.

From the paused state in a Script element you can do the following to debug the script:

- You can hover your mouse over variables in the script to see their current values. Note that if you hover over an If statement (or a While or Repeat-Until statement), it will display the statement condition (True or False). When you hover over a For or Do loop, it will display the loop counter.

  When hovering over lines to see values and debug the script, it is important to recall that a line that is outlined in black (because you have broken at that line or stepped to that line), *has not yet been evaluated*. That is, an outlined line indicates that execution is paused immediately prior to the line.

- You can press the **Resume** button or press **F5** and execution will proceed from where it is paused (until it reaches a breakpoint again).
- You can abort the model run by pressing the **Abort Run** button or pressing **Shift+F5**.
- You can turn off a breakpoint by highlighting it and pressing the **Toggle Breakpoints** button or pressing **F9**.
- You can add a new breakpoint by highlighting a line and pressing the **Toggle Breakpoints** button or pressing **F9**.
- You can press the **Step** button or press **F10** to step through execution one line at a time.

Breakpoints are not saved with the model. That is, once you close the model file, all breakpoints are intentionally removed. You will have to re-insert them if your debugging takes place over multiple GoldSim sessions.

In some cases, in order to fully understand and debug a script, you may want to print it out. You can do this from a button on the Script dialog.

Read more: Printing Scripts (page 837).

Log statements, which allow results and text to be written to the model Run Log, are also useful for debugging. These are discussed in the next section.

Logging Messages in Scripts

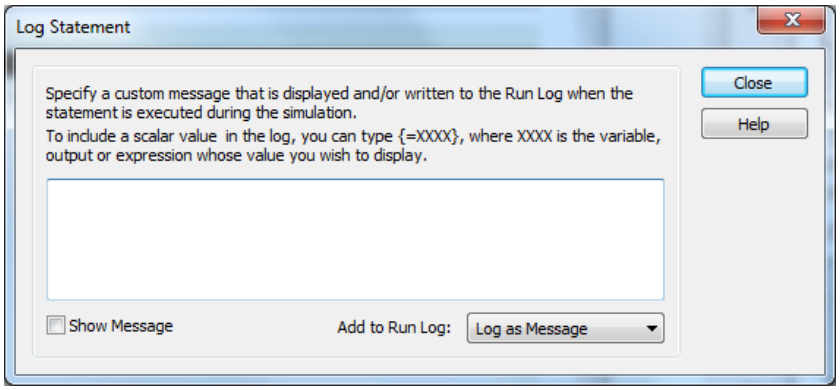
Log statements allow you to write text and computed results to the model's Run Log. This can also be useful for debugging scripts. The content can be a simple message, a warning, or a fatal error.

Read more: The Run Log (page 506).

There are three types of Log statements. All three can be inserted by selecting the appropriate type from the Script element's **Insert** menu (or pressing the appropriate hot-key when in the Script dialog):

Log Type	Hot-Key	Behavior
Message	Ctrl+L	Writes a message to the Run Log; however, no warning is displayed after the run.
Warning	Ctrl+N	Writes a message to the Run Log; a warning is displayed after the run to alert the user.
Error	Ctrl+R	Writes a message to the Run Log; treated as a fatal error (the simulation is stopped).

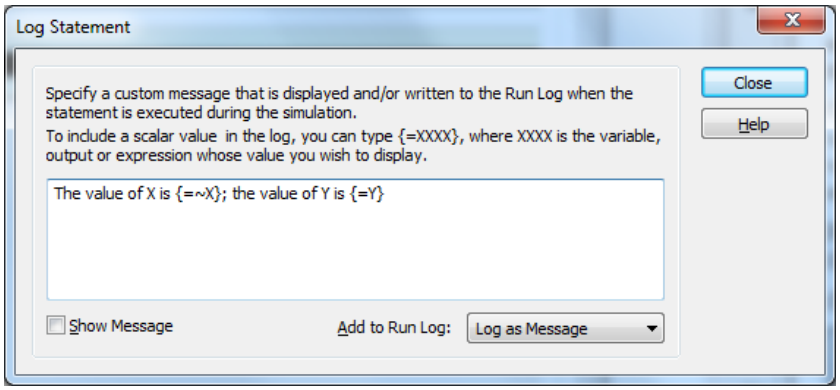
The Log statement is inserted below the statement that was highlighted at the time the new statement was selected for insertion. The following dialog will be displayed:



Note that at the bottom of the dialog you can modify the type of Log statement (Message, Warning or Error). If you check the **Show Message** checkbox, the message will also appear in a pop-up window during the simulation whenever the statement is executed.

One of the most powerful features of the Log statement is that you can write the value of a variable, output or expression in the message. To include a scalar value in the log, type {=VALUE}, where VALUE is the output, variable or expression whose value you wish to display.

For example, assume you had a script variable named X, and an external output (from outside the Script) named Y. A Log message statement like this:



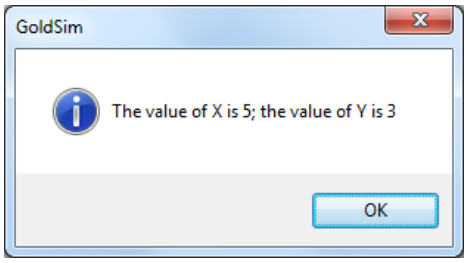
would look like this in the script:

Script

	Statement List
1	Define: X = 5
2	Log Message: The value of X is {=~X}; the value of Y is {=Y}
3	Result = 0.0

Note: When referencing script variables in a Log statement, they must be referenced using the ~ (since they are locally available properties).

The message that would be displayed in a pop-up would look like this:

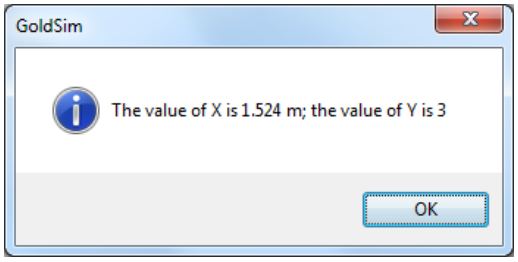


In the Run Log, it would look like this:

```
Realization 1

0 day:
\Script1: Script Message: The value of X is 5; the value of Y is 3
```

If the variables for which you are writing values have units, the units will be displayed. For example, if X was a length, this would be indicated in the message:



However, *you cannot control the units in which they are displayed*. They are always displayed in the base units for the particular dimension (e.g., meters for lengths, seconds for time).

Note: No links are created by referencing external elements in a script Log statement. Since there are no links, the causality sequence is **not** affected in any way by the existence of the expression or reference. This means that the previous (rather than current) value of an element's output could potentially be shown if the referenced element follows the Script element in the causality sequence. Usually, however, this issue will not come up, as element outputs referenced in Log statements will typically have been used elsewhere in the script (and hence would come before the Script element in the causality sequence).

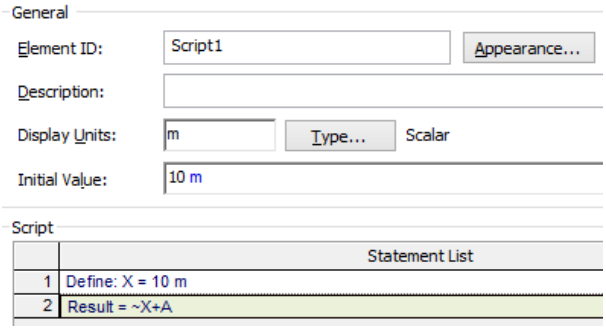
Read more: The Causality Sequence and Element Updating (page 311).

Printing Scripts

In some cases, in order to debug or modify your script, or explain it to someone else, you may want to print it out. You can do so by pressing the **Print** button on the Script dialog or by creating a text file of the Script contents using the key combination **Ctrl-Shft-Alt-S**.

When you do so, however, the printed text will not simply consist of what you would see if you were to look at the Script dialog. This is because for Variable Definition statements, there is information that is not explicitly shown in the script – the type, order and units of the variable. You must open the dialog for that statement to see this information.

Hence, when you print the script, GoldSim adds this information to the printed text. For example, consider this very simple script:



If you were to print this script, the text would look like this:

```
Script Code
-----------

Created: May 16, 2014 15:52:06
Element: Script1

Global Variables:
-----------------

// Implicit variable defining the result of the script.
VALUE[m] Result

Script:
-------

// Description for X
VALUE[m] X = 10 m
Result = ~X+A
```

Note that the text also shows the types (VALUE in this case) and units (m) for the variables, as well as descriptions. Hence, the printed script can be understood without referring back to the GoldSim model itself.

Script Examples

In order to better understand the use of Script elements, three simple example scripts are presented and briefly described in the sections below.

All three examples can be found in a file (Script.gsm) that is located in the General Examples folder of your GoldSim directory.

Script Example: Newton's Method

This simple example illustrates how the Script element can be used to carry out an iterative calculation. It can be found in a file (Script.gsm) that is located in the General Examples folder of your GoldSim directory. The example uses Newton's method to calculate the square root of 101. While you're certainly not likely to need to calculate the square root of 101 by Newton's method, the model does provide a nice example of how to iterate within a script.

Newton's method is a successive approximation method for finding real roots of differentiable functions.

For a function f(X), which in this case represents the error in the approximation, which we wish to go to zero, the formula for Newton's method is:

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)}$$

Where f ' (X_n) is the first derivative with respect to X, and X_n is the nth approximation of the root (X_0 is an initial guess for the root). Therefore, in our example of finding the square root of 101:

$f(X) = X^2 - 101$; and

$f'(X) = 2X$.

The script used for this simple example to solve for the square root of the varable Val is shown below:

Script

	Statement List
1	Define: RootGuess = Val
2	Define: MaxIterat = 10
3	FOR (I = 1; ~I <= ~MaxIterat; I = ~I + 1)
4	Define: F = ~RootGuess^2 - Val
5	Define: FPrime = 2 * ~RootGuess
6	Log Message: Iterations = {=~I}, Root = {=~RootGuess}, Error = {=~F}
7	IF (abs(~F / Val) < 1e-6) THEN
8	BREAK
9	ELSE IF (~I >= ~MaxIterat)
10	Log Error: Sq_Root failed to converge in {=~I} loops.
11	END IF
12	// Assign the next estimate for the root.
13	RootGuess = ~RootGuess - ~F / ~FPrime
14	END FOR
15	Result = ~RootGuess

The script starts out with Variable Definition statements in lines 1 and 2. Note that when script variables are referenced on the right side of equations (e.g., in line 4), a ~ symbol is required. This is because script variables are locally available properties. In line 1, the script references an element outside this Script element called Val (a constant whose square root is being sought, 101). Note that referencing another element's output does not require the ~ symbol.

Read more: Defining Local Script Variables (page 810); Assigning Values to Script Variables (page 807).

After the Variable Definition statements, a FOR loop is defined. The FOR loop starting in line 3 shows that it creates a loop variable (named "I") that is initially 1 and is incremented by 1 on each iteration, continuing while the expression "I<=MaxIterat" is true. Hence, the FOR loop executes the statements it contains until it is broken by a BREAK statement, or the loop count reaches the specified maximum number of iterations possible. In each FOR loop iteration, two more Variable Definition statements (with expressions as the assignments) compute the values for the function (F) and the derivative (FPrime). A Variable Assignment statement (line 13) then generates a new approximation of the square root at the bottom of the loop.

The IF statement block beginning at line 7 tests the approximation error in the current estimate of the square root. If it is sufficiently small, a BREAK statement is executed to terminate the FOR loop.

Read more: If-Else Statements in Scripts (page 817); Break and Continue Statements in Scripts (page 827).

If the approximation error remains high and the maximum number of iterations is reached, a LOG statement (line 10) is written to the Run Log noting that the script failed to converge. Line 6 is another LOG statement reporting the progress of the successive approximations for each iteration of the FOR loop.

Read more: Logging Messages in Scripts (page 834).

Script Example: Manipulating a Matrix

A very common application of the Script element is to construct and/or manipulate arrays. This is typically facilitated by using one of the Script element's looping constructs to assign values to rows, columns and/or individual items. The simple example described here illustrates how the Script element can be used to carry out array operations such as these. It can be found in a file (Script.gsm) that is located in the General Examples folder of your GoldSim directory.

Read more: Defining and Assigning Array Variables in a Script (page 812).

The example starts with an input matrix. In the input matrix, each column represents a variable, and each row represents an observation of the variable. The script computes the correlations among the various columns (variables) of the matrix, producing a symmetric correlation matrix (rows and columns both indexed by variable). The correlation coefficient, C_{xy}, between two columns of data (i.e., the correlation between two variables) is given by the following equation:

$$C_{xy} = \frac{\sum_{1}^{n}(x_i - m_x)(y_i - m_y)}{n s_x s_y}$$

In this equation, x and y are the variables. x_i and y_i are the i^{th} observations of variable x and y, respectively. m_x and m_y are the means of the observations for those variables (i.e., the means of all values in a particular column). s_x and s_y are the standard deviations for those observations. n is the number of observations (i.e., the number of rows).

The script used for this simple example is shown below:

Script

	Statement List
1	// Calculate standard deviations of each column
2	Define: sd[*] = sdc(Input_Matrix)
3	// Calculate the means of each column
4	Define: means[*] = meanc(Input_Matrix)
5	// Do two nested loops over the columns of the input matrix
6	DO Var2 = 1, GetColumncount(Input_Matrix), 1
7	DO Var1 = 1, GetColumnCount(Input_Matrix), 1
8	Define: temp = 0.0
9	// Do a loop up to the row count of the input matrix to sum all the squared deviations
10	DO Var3 = 1, GetRowCount(Input_Matrix), 1
11	temp = ~temp+(Input_Matrix[~Var3,~Var2]-~means[~Var2])*(Input_Matrix[~Var3,~Var1]-~me
12	END DO
13	// Finallly normalize by the product of the two column's standard deviations
14	Result[~Var1,~Var2] = ~temp/(GetRowCount(Input_Matrix)*~sd[~Var1]*~sd[~Var2])
15	END DO
16	END DO

At the top of the script (lines 2 and 4), Variable Definition statements (with expressions as the assignments) compute the means and standard deviations of each column.

Read more: Defining Local Script Variables (page 810); Assigning Values to Script Variables (page 807).

The calculation of the correlation coefficient for each column pair is then carried out using two Variable Assignment statements. The first (line 11) is in the middle of three nested DO loops. The second (line 14) is at the bottom of the second nested loop.

Read more: Do Loops in Scripts (page 821).

Script Example: Referencing a Previous Value Element in a Script

By default, a Script element has a single primary output. However, it is possible to define additional outputs for the element (with their own attributes and dimensions) when defining local variables within a script.

Read more: Understanding the Outputs of a Script Element (page 815).

There is an important difference in the behavior of the primary output and any other exposed outputs. The primary output of a Script element is a ***state variable***. A state variable provides inertia or "memory" to a system because its value is computed based on the historical value of the element's inputs (as opposed to only being a function of the current value of the element's inputs). All state variables have, by definition, an initial value. This allows the output to be computed when there are no historical inputs available (e.g., at the start of a simulation).

Read more: Understanding State Variables in GoldSim (page 309).

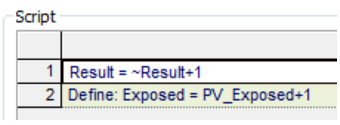
This has an important implication for how the variables can be used in a script. In particular, the value of the primary output is "remembered" between updates of the element. As a result, in a script you could create a Variable Assignment such as "Result = ~Result + 1" as the very first statement. The expression would use the *previous value* of the Result when it was evaluated. This is not possible for any of the other script variables, since they require a Variable Definition statement which initializes the variable every time the Script element is updated.

If you would like additional outputs of the Script element to be able to "remember" their previous update (and hence behave like state variables with initial conditions), you can do so by referencing the Script element's output in a Previous Value element outside of the Script element (and then referencing the Previous Value inside the script.

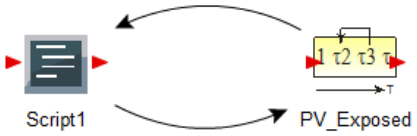
Read more: Referencing an Output's Previous Value (page 898).

This very simple example illustrates how the Script element can be used in conjunction with a Previous Value element. It can be found in a file (Script.gsm) that is located in the General Examples folder of your GoldSim directory.

The script simply updates the primary output (*Result*) and a secondary variable (*Exposed*) every time the Script element is updated (in this case, every timestep). In order for the defined variable (*Exposed*) to be "remembered" between timesteps, it is stored in a Previous Value element, and it is that element that is referenced by the script):

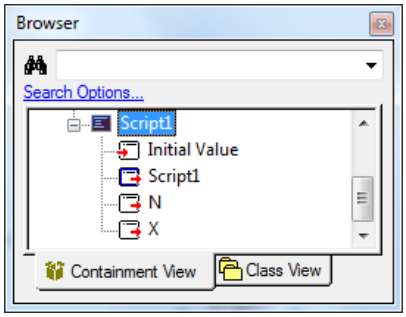


As can be seen, the script references the Previous Value element in a loop:



Browser View of a Script Element

The browser view of a Script element is shown below:



In this example, two script variables have been added as exposed outputs (N and X). By default, the Script element would only have a single output (the primary output, having the same name as the element). The Script element only has a single input – the Initial Value.

Note: Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

Read more: Using the Browser (page 110).

Using Conditional Containers

One of the advanced features in GoldSim is to make Containers conditional. Conditionality allows you to make a Container and all of its contents inactive until specific events occur and/or specific conditions are met.

Elements in an inactive container are "dormant". That is, they are not updated or recalculated each timestep, and while they are inactive their output values never change. When other specific events occur and/or conditions are met, the Container (and its contents) can become active (and hence carry out their normal calculations). A conditional Container can activate and deactivate multiple times during a simulation.

Conditionality is a very powerful feature, and can be used to

- temporarily "turn off" certain parts of your model (e.g., during a testing phase); or
- simulate processes or features which themselves only exist or are active during certain parts of your simulation.

As an example of the latter, suppose that you wished to model a facility (e.g., a chemical plant) which had a contingency plan for repairs following some event (e.g., an accident). This contingency plan could be represented within a conditional Container, and the Container could then be activated when the event occurred. The Container representing the contingency plan would operate

(accumulating costs and other consequences) until repairs were completed, and then deactivate (until another event occurred).

A simple example which illustrates the use of conditional Containers (ConditionalContainers.gsm) can be found in the Containers subfolder of the General Examples folder of your GoldSim directory. The folder also includes a more complex example of the use of conditional Containers to simulate projects (ProjectSimulation.gsm).

Behavior of Elements in Conditional Containers

Before discussing how conditional Containers can be activated and deactivated, it is useful to first explain how elements behave when they are within a conditional Container.

Elements within conditional Containers behave as follows:

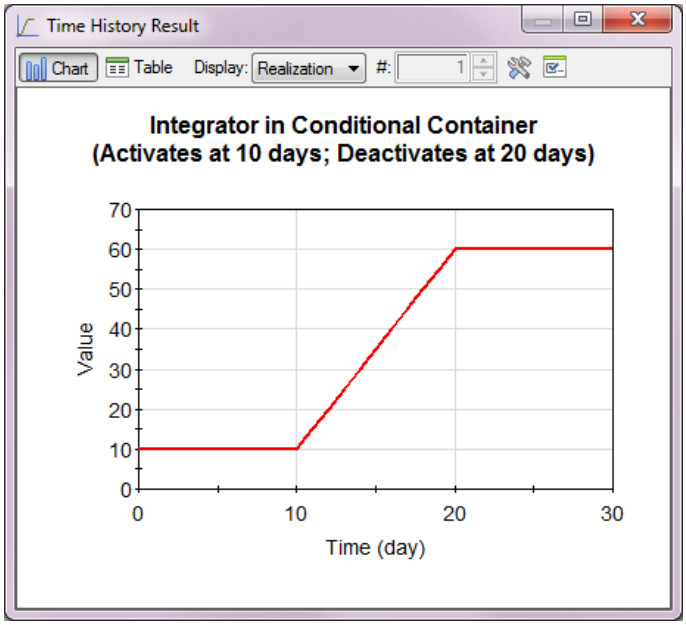
- Before a Container is activated for the first time in a realization, any elements it contains behave as follows:
  - Data elements and Expressions that are static (i.e., have constant inputs or are functions of elements with constant inputs) are computed at the beginning of each realization.
  - Static Stochastic elements (Stochastics whose inputs are constant, and have no triggers) are sampled at the beginning of each realization.
  - Conditional Containers have no effect on most Time Series elements, which always output their results. Time Series elements that are set to *record*, however, cannot be placed inside a Conditional Container that can be deactivated.
  - Outputs with clearly defined non-zero initial values (i.e., Integrators, Reservoirs, Material and Information Delays, Previous Value elements, Status elements, and Milestones) take on their initial values at the beginning of each realization.
  - All other outputs are set to zero.
- Once a Container is activated, all the elements that it contains output their normal (computed) output.
- After a Container is deactivated, any elements it contains behave as follows:
  - Most outputs that are continuous (as opposed to discrete) continue to transmit their last active value.
  - The exception to this are outputs that represent material flows (e.g., the output of a Material Delay; the Overflow Rate from a Reservoir). These outflows are set to zero.
- While a Container is inactive,
  - All incoming material flows (e.g., Rates of Addition to Reservoirs, Flows into Material Delays) or discrete changes of material are ignored and discarded. Warning messages are written to the Run Log whenever material flows from active elements (outside the conditional Container) to inactive elements are discarded.
  - Any other discrete signals sent to inactive elements are ignored and discarded. Warning messages are written to the Run Log when discrete additions or withdrawals of material flows to inactive elements are discarded.

- Any timed events or events triggered using an On Event trigger are ignored and discarded.
- Any discrete or continuous signals or flows that are in transit within Delays are "frozen" (the conveyor is stopped).
- If an inactive Container is activated at exactly the same as some child elements are triggered, the triggers to the children are not ignored (they are applied immediately upon activation of the Container).

Warning: On Changed, On True and On False events are triggered whenever the element determines that the argument to the event has changed, become true, or become false, respectively. If the element is in a conditional Container, it does not evaluate these arguments while it is inactive, and therefore some On Changed, On True and On False events that occur while the Container is inactive can be "delayed" until the element activates (rather than ignored). For example, if an On True trigger is defined as "Etime > 15 days", and the element is inactive until 20 days, the event will actually be triggered at 20 days (since, from the element's viewpoint, this is first time that it is able to determine that the condition has become true).

As a simple example of the behavior of an element inside a conditional Container, consider an Integrator with an Initial Value of 10, and a Rate of Change of 5 per day. Assume that the Integrator was within a conditional Container that was initially inactive, was activated at 10 days, and deactivated at 20 days.

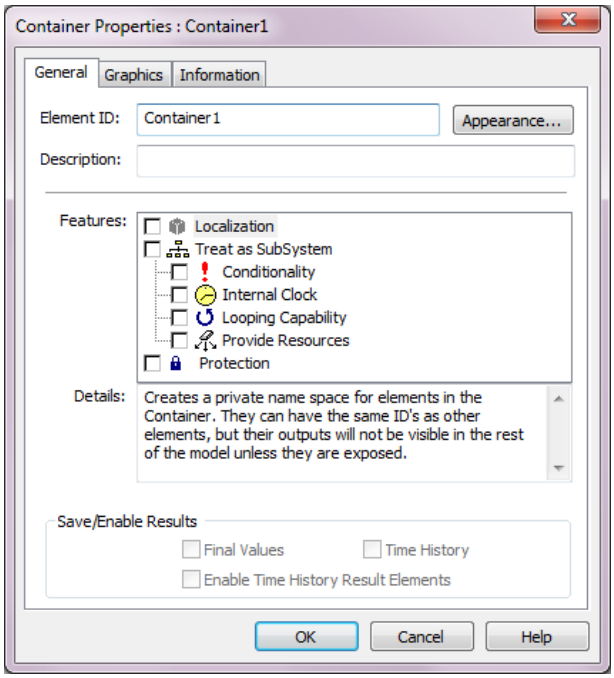
In this case, the output of the Integrator would behave as shown below:



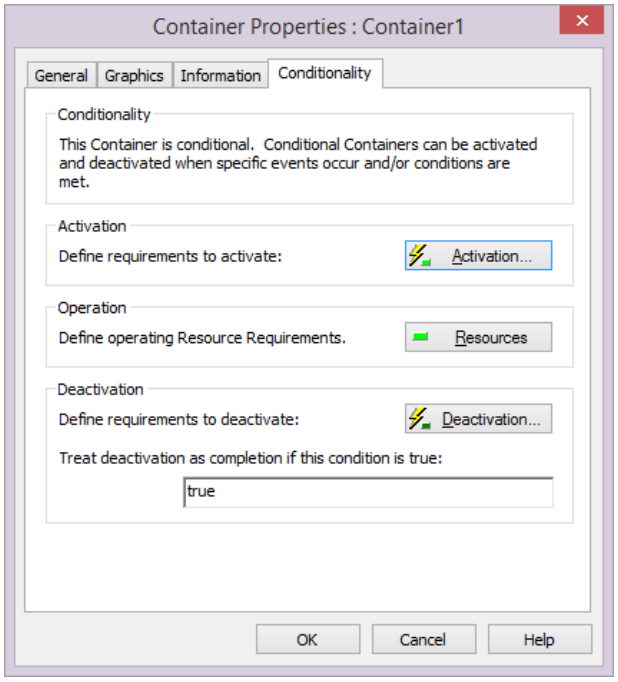
The Integrator starts at 10. It remains at 10 until the Integrator activates at 10 days. Between 10 and 20 days, the output increases linearly (to 60), and after 20 days, the output stays constant at 60.

Enabling and Disabling Conditionality

Double-clicking on any Container brings up the Container's property dialog:

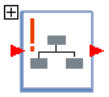


Containers can be made conditional by selecting the **Conditionality** feature in the Container dialog. When you do so, a **Conditionality** tab is added to the Container dialog:



You can turn off conditionality by selecting the **Conditionality** feature in the Container dialog to clear the box. When you do so, the **Conditionality** tab is removed.

Conditional Containers are represented in the graphics pane as follows:



Warning: When you make a Container conditional, the **Treat as SubSystem** feature is also automatically selected (and cannot be deselected unless you first turn off **Conditionality**). That is, a Conditional Container, by definition, is treated as a SubSystem. Because a Conditional Container is treated as a SubSystem, this puts certain limitations on how Conditional Containers can be used.

Read more: Treating a Container as a SubSystem (page 139).

Outputs of a Conditional Container

When you enable conditionality for a Container, seven outputs are added to the Container. While a regular Container (i.e., one which is not conditional) has no outputs of its own, the seven outputs of a conditional Container are properties of the Container itself.

Four of the outputs are values or conditions:

Activity_Status: This is a condition which is True when the Container is active.

Completion_Status: This is a condition which is True when the Container has deactivated and the Container is considered to have been *completed*. A conditional Container is considered to have completed if, at the time it deactivates, the expression in the field named **On deactivation, output Completion event if this condition is true** evaluates to True. If a Container reactivates after completing, the Completion_Status is reset to False.

Read more: Deactivating a Container (page 847).

Num_Activations: This is the cumulative number of times that the Container has been activated.

Duration: This represents the duration that the Container has been Active.

If you wish to reference any of these outputs *outside* of the Container, you must do so by referencing the Container name as a prefix (e.g., Container1.Duration).

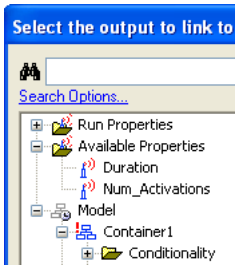
You cannot reference the **Activity_Status** or the **Completion_Status** of a Container inside a Container (GoldSim will say that this will create a recursive system). There is, in fact, no reason to do so, since if an element in the Container is being updated, the Container must be active (so that the Activity_Status must be True and the Completion_Status must be False).

In some cases, you may want to reference the **Duration** and/or **Num_Activations** outputs inside the Container or when defining Deactivation triggers for the Container. GoldSim does support this. However, you cannot do so by using the Container name (i.e., Container1.Duration). Instead, you must reference these two variables using the prefix "~" (e.g., ~Duration and ~Num_Activations). As indicated by the way they are referenced, these two outputs are examples of locally available properties.

Read more: Understanding Locally Available Properties (page 750).

If you access the Insert Link dialog *inside* a conditional Container (by right-clicking inside an input field of an element within the Container or within the Deactivate trigger dialog for the Container), these two outputs appear in an

Available Properties folder (displaying all of the locally available properties that can be accessed from that location):



Conditional Containers also have three additional outputs which are events:

Activation_Event: This is an event that is output when the Container is activated. If a Container is already active when an Activation trigger is received, it *will not* output another Activation_Event at that time.

Read more: Activating a Container (page 846).

Completion_Event: This is an event which is output when the Container has deactivated and the Container is considered to have been *completed*. A conditional Container is considered to have completed if, at the time it deactivates, the expression in the field named **On deactivation, output Completion event if this condition is true** evaluates to True.

Read more: Deactivating a Container (page 847).

Termination_Event: This is an event which is output when the Container has deactivated and the Container is considered *not* to have been *completed*. A conditional Container is considered to have *not completed* if, at the time it deactivates, the expression in the field named **On deactivation, output Completion event if this condition is true** evaluates to False.

Activating a Container

You control when a conditional Container activates via the **Activation…** button on the **Conditionality** tab, which provides access to a standard Trigger dialog.

Read more: Understanding Event Triggering (page 323).

By default, the Activation is set to "Never". This means that the Container will never activate.

Note: You can tell if an activation trigger has been defined from the appearance of the **Trigger…** button. If a trigger is defined, the rectangle next to the lightning bolt is bright green; otherwise it is dark green. And like all **Trigger…** buttons, it displays a tool-tip. If there is no trigger defined, the tool-tip will display "Never activates".

You can explicitly control when a conditional Container activates by defining one or more triggers (e.g., "Activate when Time > 10 days").

Whenever a conditional Container is activated, an Activation_Event output is released.

Read more: Outputs of a Conditional Container (page 845).

Note: You can record all activation and deactivation times in the model's Run Log. This logging can be activated via the **General** tab of the Options dialog (accessed via **Model|Options** from the main menu).

Read more: The General Tab of the Options Dialog (page 408).

If you wish the Container to activate when its parent Container activates, set the Activation trigger to **Auto Activate**. Note that if a conditional Container is set to **Auto Activate** and it is not within a conditional Container, it activates when the Model (root) Container activates (i.e., at the start of the realization).

Several important points should be noted regarding activation of Containers:

- Even if the Container is activated via a trigger, it can only activate if its parent Container is active. If its parent Container is inactive, activation triggers will be ignored (and discarded).
- A Container can be activated multiple times during a realization. Of course, in order for a Container to be reactivated, it must first be deactivated. Activation triggers that occur while the Container is active are ignored (they are not "stored" or saved).
- If an inactive Container is activated at exactly the same time as some child elements are triggered, the triggers to the children are not ignored (they are applied immediately upon activation of the Container).

One common use of conditional Containers is to use the activation of a Container to simultaneously trigger multiple elements within the Container.

For example, if you wished to trigger a number of Discrete Change elements and/or resample a number of Stochastics, you could do so by placing them inside a conditional Container *and* specifying (in their Trigger dialogs) that they are automatically triggered (**Auto Resample** for Stochastics and **Auto Trigger** for Discrete Changes). When the Container was triggered to activate, the Stochastics would automatically be resampled, and the Discrete Changes would automatically be triggered.

Note: You cannot reference the locally available properties Duration or Num_Activations in an Activation trigger. Arguments to an Activation trigger must come from outside of the Container.

Deactivating a Container

Once a Container is activated, it can be deactivated via the **Deactivation…** button on the **Conditionality** tab, which provides access to a standard Trigger dialog.

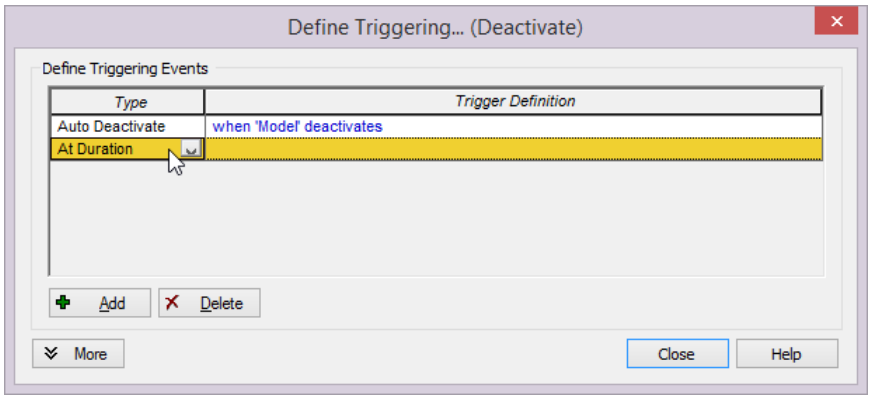
Read more: Understanding Event Triggering (page 323).

You can explicitly control when a conditional Container deactivates by defining one or more triggers.

By default, the Deactivation is set to **Auto Deactivate**. This means that the Container will deactivate when its parent Container deactivates. Note that if a conditional Container is not within a conditional Container, it deactivates when the Model (root) Container deactivates (i.e., at the end of the realization).

Note: You can tell if a deactivation trigger has been defined from the appearance of the **Trigger…** button. If a trigger is defined, the rectangle next to the lightning bolt is bright green; otherwise it is dark green. And like all **Trigger…** buttons, it displays a tool-tip. If there is no trigger defined, the tool-tip will display "Automatically deactivates when *parent* deactivates", where parent is the name of the parent Container.

One special trigger type is provided within the Deactivation trigger dialog:



The **At Duration** trigger deactivates the Container when the Duration output exceeds the specified Trigger Definition (which must represent a length of time). The Duration output represents the amount of time that the Container has been active. Hence, this provides a convenient mechanism to deactivate a Container once it has been active for a specified amount of time. The Trigger Definition input in this case would typically be defined by an element inside the Container.

Read more: Outputs of a Conditional Container (page 845).

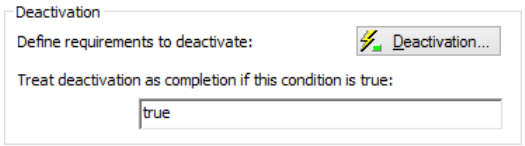
Note: You can record all activation and deactivation times in the model's Run Log. This logging can be activated via the **General** tab of the Options dialog (accessed via **Model|Options** from the main menu).

Read more: The General Tab of the Options Dialog (page 408).

Several points should be noted regarding deactivation of Containers:

- A Container can be deactivated multiple times during a realization. Of course, in order for a Container to be deactivated a second time, it must first be reactivated. Deactivation triggers which occur while the Container is inactive are ignored (they are not "stored" or saved).
- All conditional Containers have an Auto Deactivate trigger (that cannot be deleted). Hence, it will always immediately deactivate if its parent Container deactivates.
- A conditional Container can be activated and immediately deactivated by assigning the same trigger for Activation and Deactivation.

When a conditional Container is deactivated, one of two possible events are released, depending on the condition specifed in the **Treat as completion if this condition is true** field (which defaults to True):



Completion_Event: This event is output when the Container has deactivated and the Container is considered to have been *completed*. A conditional Container is considered to have completed if, at the time it deactivates, the expression in the field named **On deactivation, output Completion event if this condition is true** evaluates to True.

Termination_Event: This event is output when the Container has deactivated and the Container is considered *not* to have been *completed*. A conditional Container is considered to have *not completed* if, at the time it deactivates, the expression in the field named **On deactivation, output Completion event if this condition is true** evaluates to False.

Note: If a Container is deactivated due to an operating Resource Requirement not being met, neither a Completion_Event nor a Termination_Event is fired.

Read more: Specifying Resources for a Conditional Container (page 849).

Deactivating a Container and differentiating whether it has terminated or completed is particularly useful for simulating projects (i.e., when you are using the conditional Container to simulate a project or a task). In this case, you often need to track whether or not various tasks (or collections of tasks) have been successfully completed, as this may impact how subsequent tasks are carried out.

Using Auto Triggers in Conditional Containers

Most elements with triggers can be assigned an Auto Trigger. An Auto Trigger requires no user-defined Trigger Definition and its behavior is defined by its context (i.e., the type of element). Auto Triggers react to the activation or deactivation of their parent Container.

Read more: Understanding Event Triggering (page 323).

In most cases, an Auto Trigger is used in association with the activation of a conditional Container. In particular, if an element inside a conditional Container is given an Auto Trigger, it will be triggered when its parent Container activates.

In one instance, an Auto Trigger is associated with the deactivation of a Container. In particular, conditional Containers all have **Auto Deactivate** triggers (that cannot be deleted). This means that the Container will deactivate when its parent Container deactivates. Note that if a conditional Container is not within a conditional Container, it deactivates when the Model (root) Container deactivates (i.e., at the end of the realization).

Specifying Resources for a Conditional Container

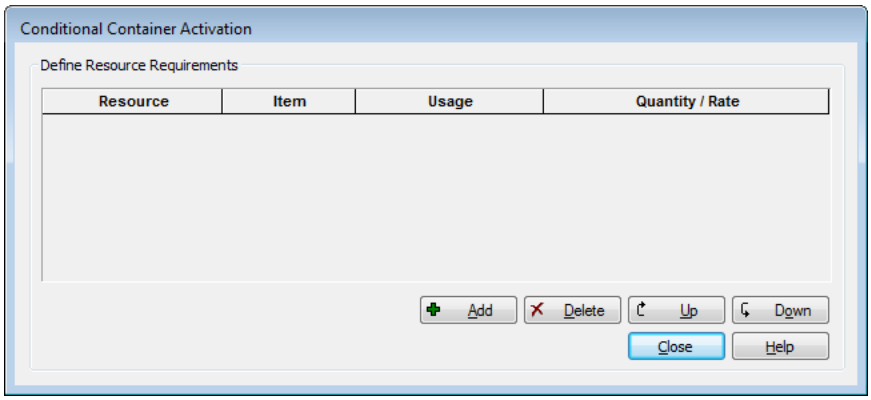
Conditional Containers can have specified Resource Requirements.

Read more: Using Resources (page 781).

There are two types of Resource Requirements that can be specified for Conditional Containers:

- Requirements to activate the Container;
- Requirements for the Container to operate.

To define a Resource Requirement for activating a Conditional Container, press the **Resources…** button in the **Activation…** trigger dialog for the element. The following dialog will be displayed:



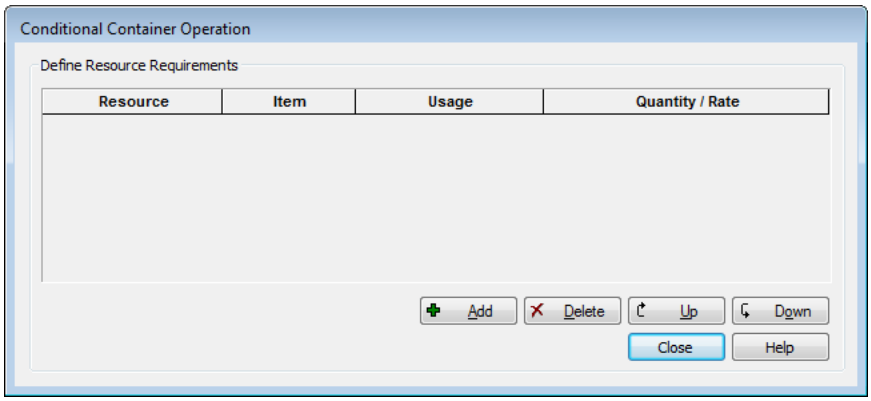
You can add a Resource Requirement by pressing the **Add** button.

Read more: Interacting with Resources (page 790).

A Conditional Container Activation trigger interacts with the specified Resource Stores when the Container is activated, and can only have three types of interactions (specified in the **Usage** column):

- **Spend (discrete):** A discrete quantity of the Resource is required in order to activate the Container. If triggered with a Spend Requirement and the requested Resource quantity is not available, the trigger is held (it waits) until the Resource becomes available. If the Resource becomes available at a later time in the simulation, this creates a delay in the activation of the Container; if the Resource never becomes available, the Container is never activated.
- **Borrow (discrete):** A discrete quantity of the Resource is required in order to activate the Container. If triggered with a Borrow Requirement and the requested Resource quantity is not available, the trigger is held (it waits) until the Resource becomes available. If the Resource becomes available at a later time in the simulation, this creates a delay in the activation of the Container; if the Resource never becomes available, the Container is never activated. Once the Container is deactivated, the borrowed quantity is returned to the Resource Store.
- **Deposit (discrete):** A discrete quantity of the Resource is created and deposited with the Store when the Container is activated.

To define a Resource Requirement for operating a Conditional Container, press the **Resources…** button on the **Conditionality** tab of the Conditional Container (labeled "Define operating Resource Requirements"). The following dialog will be displayed:



You can add a Resource Requirement by pressing the **Add** button.

Read more: Interacting with Resources (page 790).

A Conditional Container Operation Requirement interacts with the specified Resource Stores when the Container is activated, and can only have two types of interactions (specified in the **Usage** column):

- **Spend At Rate:** A specified continuous spend rate of the Resource is required in order for the Container to remain Active. If the requested Resource is not available in sufficient quantity to maintain the requested spend rate, the Conditional Container is deactivated. If it becomes available again, the Container reactivates.

Note: If a Container is deactivated due to an operating Resource Requirement not being met, although the Activity_Status is set to False, neither a Completion_Event nor a Termination_Event is fired.

Read more: Outputs of a Conditional Container (page 845).

Note: If a Container is deactivated due to an operating Resource Requirement not being met, it will reactivate if at least one timestep's worth of the Resource requirement (i.e., a quantity equal to Spend At Rate * Timestep Length) becomes available.

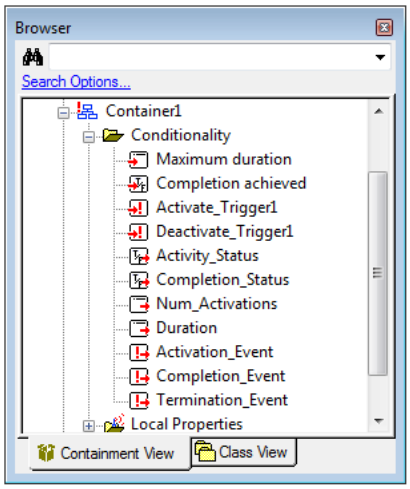
- **Deposit At Rate:** While the Container is active, the specified deposit rate is added to the Store.

Resources are an advanced feature, and you should read the section listed below before attempting to use them.

Read more: Using Resources (page 781).

Viewing a Conditional Container in the Browser

The browser view of a conditional Container is shown below:



In addition to presenting the Container's contents, a "Conditionality" folder is added to the Container's browser view. This folder contains the input triggers for activation and and other inputs, and the outputs of the Container.

Note: Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

Read more: Using the Browser (page 110).

Using External Application Elements

GoldSim provides three elements that can interact with external applications:

- Spreadsheet elements;
- External (DLL) elements; and
- File elements.

These are discussed below.

Spreadsheet Elements

A Spreadsheet element allows you to dynamically exchange data with a Microsoft Excel spreadsheet file. You can transmit data into specified cells in the spreadsheet, have the spreadsheet recalculate, and copy data from "result" cells back into your GoldSim model. Alternatively, you can simply use a spreadsheet as a source of input data to GoldSim, or as a repository for output data.

Note: If you wish to import lookup tables or time history data from a spreadsheet into GoldSim, or export time history data from GoldSim to a spreadsheet, GoldSim provides specialized ways to do this using the Lookup Table, Time Series and Time History Result elements that do not require a Spreadsheet element.

Read more: Linking a Lookup Table to a Spreadsheet (page 274); Importing Data into a Time Series from a Spreadsheet (page 194); Exporting from a Time History Result Element to a Spreadsheet (page 678).

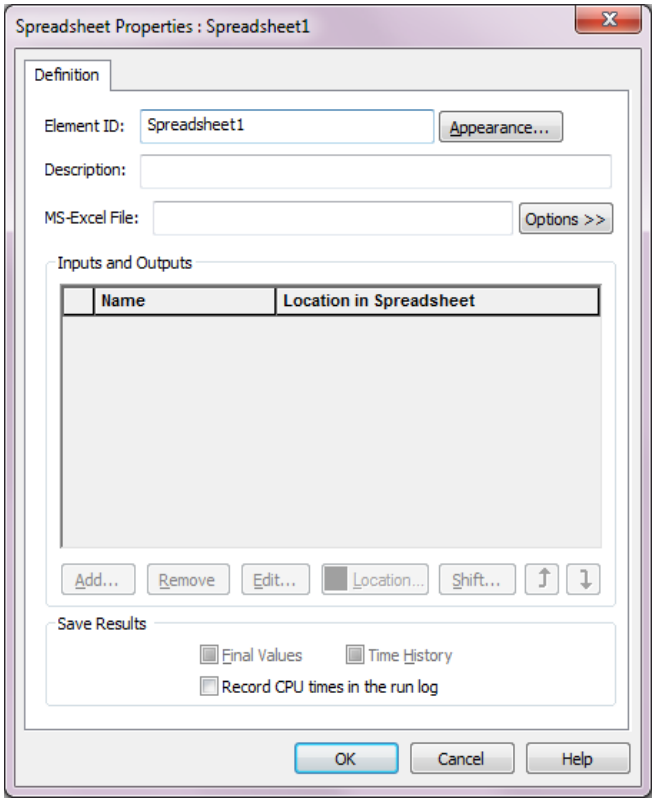
Note: GoldSim supports .xlsx, .xlsm, and .xls Excel files. However, if you have an older version of Excel (prior to Office 2007), you will need to install Microsoft's Office Compatibility Pack in order to read .xlsx and .xlsm files. Excel 2007 and later support a larger worksheet size (1,048,576 rows by 16,384 columns) than earlier versions (65,536 rows by 256 columns). If you wish to exchange data between an extended worksheet range and GoldSim, you must use Excel 2007 or newer, and the file format must be .xlsx or .xlsm. Note that GoldSim does not officially support versions of Excel prior to Excel 2003.

Warning: You should never try to open a spreadsheet that is linked to GoldSim while a simulation is running. In addition, prior to running a GoldSim model that is linked to a spreadsheet, all referenced spreadsheets should be closed. If you try to interact with a spreadsheet that is being used by GoldSim during a simulation, all interaction with the spreadsheet is blocked by GoldSim. However, under some circumstances this may not be possible and could lead to errors. Hence, as a general rule, you should not use Excel while a Goldsim model that references Excel is running.

An example model which uses a Spreadsheet element (Spreadsheet.gsm) can be found in the General Examples folder in your GoldSim directory.

Defining the Properties of a Spreadsheet Element

The properties dialog for a Spreadsheet element looks like this:



You must first enter the name of a Microsoft Excel spreadsheet file to which you wish to link by pressing the **Options >>** button. This will provide options for either selecting an existing file, or creating (and then selecting) a new file.

Note: If you select a file in the same directory as (or a subdirectory below) your GoldSim .gsm file, GoldSim will subsequently display just a local path. If you select a file in a directory above your .gsm file, it will display the full path.

Once you have selected a file, you can subsequently use the **Options >>** button to select a different file. You can also use the **Options>>** button to open the selected file in Excel. (Note that multiple Spreadsheet elements can link to the same spreadsheet file.)

Spreadsheet elements can be used in one of three ways:

1. You can use the spreadsheet as a database and import data from the spreadsheet into GoldSim at the beginning of the simulation.
2. You can use the spreadsheet as a repository for results and export data from GoldSim to the spreadsheet throughout a simulation.
3. You can use the spreadsheet as a function by exporting data into specified cells in the spreadsheet from GoldSim, having the spreadsheet recalculate, and then import data from "result" cells in the spreadsheet back into your GoldSim model.

In all cases, you need to define either at least one input to the spreadsheet element (data that you are exporting from GoldSim to the spreadsheet), or one output from the spreadsheet (data that you are importing from the spreadsheet into GoldSim).

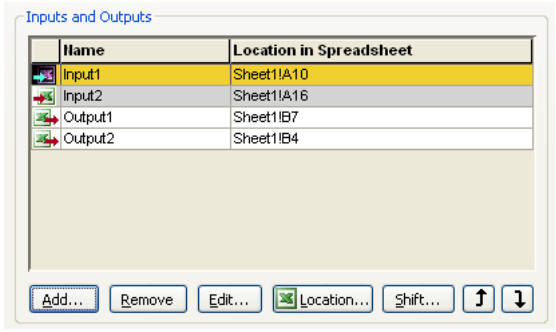
The dialog lists all of the inputs and outputs that you have specified (inputs are listed in the table first).

You add inputs and outputs by pressing the **Add…** button (which only becomes available after you have specified a spreadsheet file). After pressing the **Add…** button, the following dialog is displayed:

You must select whether you want to Import data from the spreadsheet (creating an output to the element) or Export data to the spreadsheet (creating an input for the element). If **Use Wizard to define spreadsheet inputs and outputs** is checked, a wizard will pop up to guide you through the process of creating an input or output. Otherwise, a dialog will appear. It is recommended that you use the wizard until you are familiar with the use of the Spreadsheet element.

Read more: Using the Spreadsheet Wizard to Define Spreadsheet Inputs (page 856); Using the Spreadsheet Wizard to Define Spreadsheet Outputs (page 863).

Once you have defined or edited an input or output, the main Spreadsheet element dialog is displayed again, listing all of the inputs and outputs that you have specified (inputs are listed in the table first):



An input or output can be edited by either selecting the row and pressing **Edit…**, or by double-clicking on the row. You can edit the spreadsheet location of any entry by selecting the row and pressing **Location…**. An input or output can be deleted by pressing the **Remove** button.

The up and down arrow buttons move the entries up and down the list (although inputs will always be listed before outputs). Note that the relative positions of the rows have no effect on how the element functions, and such repositioning is purely cosmetic.

The **Shift…** button provides access to a dialog that allows you to quickly edit the spreadsheet location of one or more inputs or outputs by shifting them by one or more sheets, rows or columns.

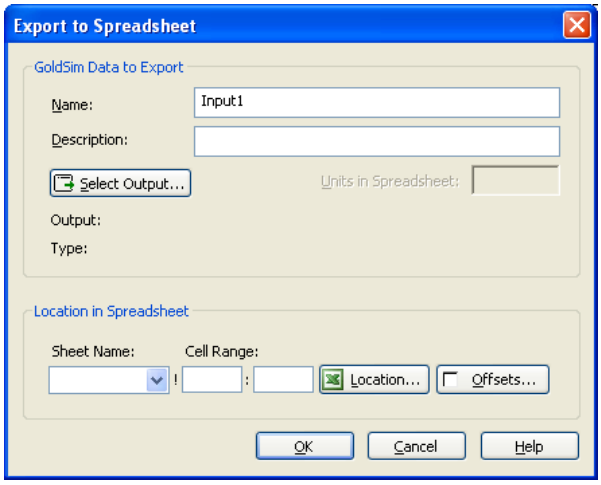
Read more: Shifting Ranges for Inputs and Outputs to a Spreadsheet Element (page 867).

If you check the **Record CPU times in the run log** box, if the element uses more than 1 CPU seconds, a message will be written to the GoldSim run log identifying the element's name, type (i.e., Spreadsheet), the number of times it was updated, and the total CPU time used.

Read more: The Run Log (page 506).

Spreadsheet Element Inputs: Exporting Data to the Spreadsheet

If you press the **Export…** button from the Import or Export Selection dialog (and the checkbox for using the wizard is cleared), or if you select an existing input in the main Spreadsheet dialog and press the **Edit…** button, the following dialog is displayed:



This dialog is used to create an input to the Spreadsheet element. An input to a Spreadsheet element is defined as an existing GoldSim output that you wish to export from GoldSim to the spreadsheet file.

You define an input as follows:

1. Select the GoldSim output that you wish to export to the spreadsheet by pressing the **Select Output…** button, which will display a browser for selecting an output. You can select a scalar, a vector or a matrix. The output can be a Value or a Condition. True Conditions will export into the spreadsheet as 1; False Conditions will export into the spreadsheet as 0.
2. The **Units in Spreadsheet** will default to the display units for the output that you selected. If the units of the data in the spreadsheet are different from this, you should change the units accordingly.
3. The **Name** of the input will default to Input*n*, where n is an integer. This name is used to label the inputs to the Spreadsheet element in the browser and the element's input interface. If you wish to, you can change this name, but it is not required. The optional **Description**

appears only in this dialog and is useful for helping to document your spreadsheet link.

4. You must then select the location in the spreadsheet to which you wish to export the GoldSim data. You can do this by specifying the Sheet Name and Cell Range directly. Alternatively, you can press the **Location…** button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell range using your mouse.

Note: The size of the range must be consistent with the dimensions of the input which you are sending to the spreadsheet. Hence, a scalar input must map to a single cell; a vector input must map to a single column or row range with the number of rows or columns matching the number of items in the vector; and a matrix input must map to a rectangular range with the number of rows and columns matching the dimensions of the matrix (e.g., a matrix with 3 rows and 6 columns would have to map to a range in the spreadsheet 3 rows long and 6 columns wide).

If required, you can control the sheet and cell range into which GoldSim exports data dynamically. For example, you could instruct GoldSim to change the location to which it exports data based on the simulation time or the realization.

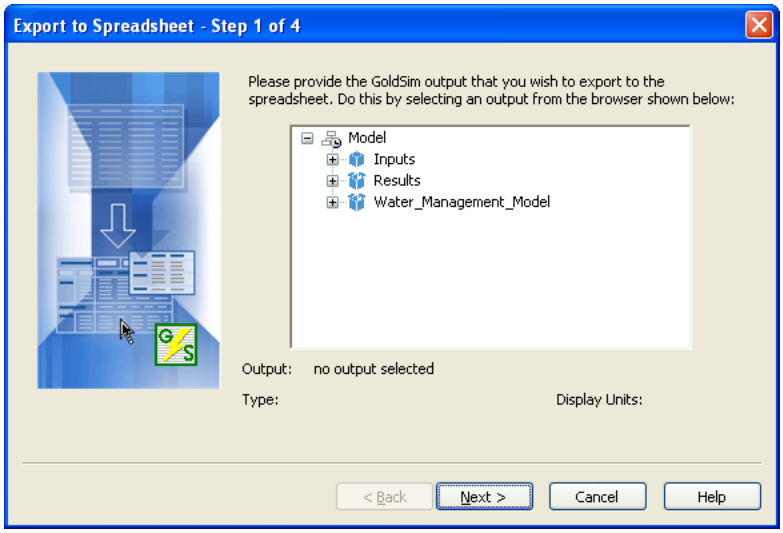
Read more: Defining Offsets for Inputs and Outputs to a Spreadsheet Element (page 865).

If you would like to edit the location of several inputs (or outputs) simultaneously by shifting them by sheet, row or column, you can do this from the main Spreadsheet element dialog using the **Shift…** button.

Read more: Shifting Ranges for Inputs and Outputs to a Spreadsheet Element (page 867).

Using the Spreadsheet Wizard to Define Spreadsheet Inputs

If you press the **Export…** button from the Import or Export Selection dialog, and the checkbox for using the wizard is checked, the Spreadsheet Wizard will be displayed:



This wizard is used to create an input to the Spreadsheet element. An input to a Spreadsheet element is defined as an existing GoldSim output that you wish to export from GoldSim to the spreadsheet file. Until you become comfortable

creating inputs and outputs for Spreadsheet elements, it is recommended that you use the wizard.

The wizard has four pages:

1. On the first page you must select the GoldSim output that you wish to export to the spreadsheet. The wizard displays a browser for selecting an output. You can select a scalar, a vector or a matrix. The output can be a Value or a Condition. True Conditions will export into the spreadsheet as 1; False Conditions will export into the spreadsheet as 0.
2. On the second page, you specify the **Name** of the input. It will default to Input*n*, where n is an integer. This name is used to label the inputs to the Spreadsheet element in the browser and the element's input interface. If you wish to, you can change this name, but it is not required. The optional **Description** appears only in the dialog for subsequently editing this Input and is useful for helping to document your spreadsheet link.
3. On the third page, you must specify the **Units in Spreadsheet.** They will default to the display units for the output that you selected. If the units of the data in the spreadsheet are different from this, you should change the units accordingly. Note that this page is skipped if the output is dimensionless or a Condition.
4. On the last page of the wizard, you select the location in the spreadsheet to which you wish to export the GoldSim data. You can do this specifying the Sheet Name and Cell Range directly. Alternatively, you can press the **Location…** button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell range using your mouse.

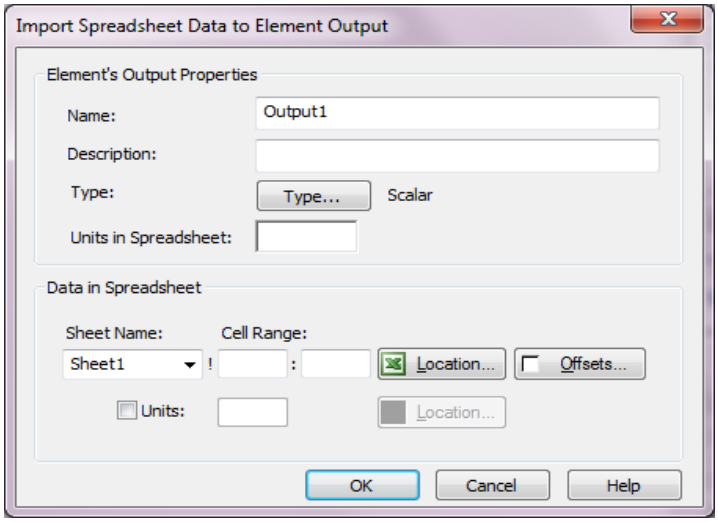
Note: The size of the range must be consistent with the dimensions of the input which you are sending to the spreadsheet. Hence, a scalar input must map to a single cell; a vector input must map to a single column or row range with the number of rows or columns matching the number of items in the vector; and a matrix input must map to a rectangular range with the number of rows and columns matching the dimensions of the matrix (e.g., a matrix with 3 rows and 6 columns would have to map to a range in the spreadsheet 3 rows long and 6 columns wide).

If required, you can control the sheet and cell range into which GoldSim exports data dynamically. For example, you could instruct GoldSim to change the location to which it exports data based on the simulation time or the realization.

Read more: Defining Offsets for Inputs and Outputs to a Spreadsheet Element (page 865).

Spreadsheet Element Outputs: Importing Data from the Spreadsheet

If you press the **Import…** button from the Import or Export Selection dialog (and the checkbox for using the wizard is cleared), or if you select an existing output in the main Spreadsheet dialog and press the **Edit…** button, the following dialog is displayed:

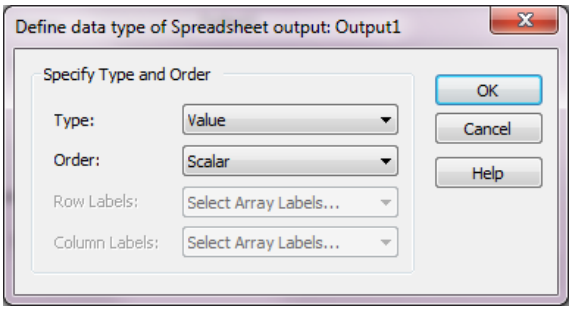


This dialog is used to create an output to the Spreadsheet element. An output to a Spreadsheet element is defined by specifying a cell range in the spreadsheet file that you wish to import from the spreadsheet into GoldSim, making the data accessible within GoldSim as an element output.

You define an output as follows:

1. The **Name** of the output will default to Output*n*, where n is an integer. You will use this name to reference the output within GoldSim (as SpreadsheetElementName.OutputName). Hence, you will likely want to change this name from its default. The optional **Description** appears only in this dialog and is useful for helping to document your spreadsheet link.

2. Press the **Type…** button to define the attributes of the output. The following dialog will be displayed:



   This dialog allows you to define the Order (scalar, vector, matrix) of the output. You can also define the Type. By default, the Type is a Value (conditions are not supported and cannot be selected here). However, you can also select Probability Distribution, which allows you to import a series of input fields that define a distribution.

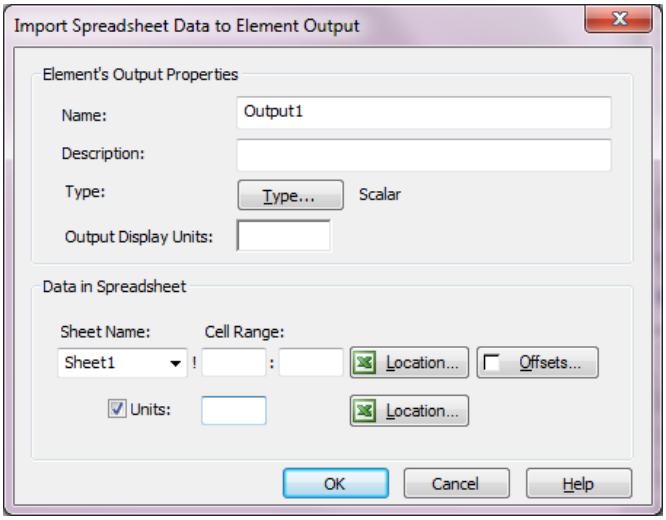
   Read more: Importing Stochastic Element Definitions from a Spreadsheet (page 860).

3. You must then specify the units of the date being imported from the spreadsheet. There are two ways to do this. You can simply specify in this dialog what the units are (and this is the default). In this case, the **Units in Spreadsheet** represents the units of the data in the spreadsheet. This will also become the display units for the output after

it is imported into GoldSim. As discussed below, alternatively, you can also import the unit string directly from the spreadsheet.

4. You must then select the location in the spreadsheet from which you wish to import the data. You can do this by specifying the Sheet Name and Cell Range directly. Alternatively, you can press the **Location…** button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired Cell Range using your mouse.

5. You can optionally choose to import the units for the data directly from the spreadsheet (rather than specifying the units directly as in step 3 above). In this case, select the **Units** checkbox in the dialog:



You must then specify the Cell containing the unit directly to the right of the **Units** checkbox (or alternatively, you can press the **Location…** button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired Cell using your mouse). Note that when you check this box, the field directly below the Type button changes from **Units in Spreadsheet** to **Output Display Units**. This allows the output to have different display units than the units in the spreadsheet (although, of course, they must be of the same dimension).

Note: The size of the range specified in the spreadsheet must be consistent with Order (scalar, vector, matrix) that you defined for the output. Hence, a scalar output must map to a single cell; a vector output must map to a single column or row range with the number of rows or columns matching the number of items in the vector; and a matrix output must map to a rectangular range with the number of rows and columns matching the dimensions of the matrix (e.g., a matrix with 3 rows and 6 columns would have to map to a range in the spreadsheet 3 rows long and 6 columns wide). When importing a Probability Distribution, the range is a single cell (the first cell in the string of inputs defining the distribution)

Note: Spreadsheets cells can be defined as TRUE or FALSE. If you try to import such a cell into GoldSim, a TRUE cell will be imported as the number 1, and a FALSE cell will be imported as 0.

If required, you can control the sheet and cell range from which GoldSim imports data dynamically. For example, you could instruct GoldSim to change the location from which it imports data based on the simulation time or the realization.

Read more: Defining Offsets for Inputs and Outputs to a Spreadsheet Element (page 865).

If you would like to edit the location of several inputs (or outputs) simultaneously by shifting them by sheet, row or column, you can do this from the main Spreadsheet element dialog using the **Shift…** button.

Read more: Shifting Ranges for Inputs and Outputs to a Spreadsheet Element (page 867).

Note: Prior to running the model, you can view any spreadsheet data that you plan to import by first selecting the **Update Spreadsheet Outputs** item in the menu displayed via the **Options>>** button for the Spreadsheet element, and then viewing the outputs of the element in tool-tips within the element's output port.

Importing Stochastic Element Definitions from a Spreadsheet

You can use the Spreadsheet element to import a probability distribution from a spreadsheet. The distribution is defined in the spreadsheet using a number of fields that specify the type of distribution, as well as the arguments to the distribution.

Probability distributions must be specified in the spreadsheet in a very specific format. In particular, the distribution must be defined in a row of adjacent cells in the spreadsheet. The first cell is a code that defines the type of distribution. The subsequent cells (to the right of the code) define the input parameters for the distribution. The table below specifies how each type of distribution must be entered in the spreadsheet.

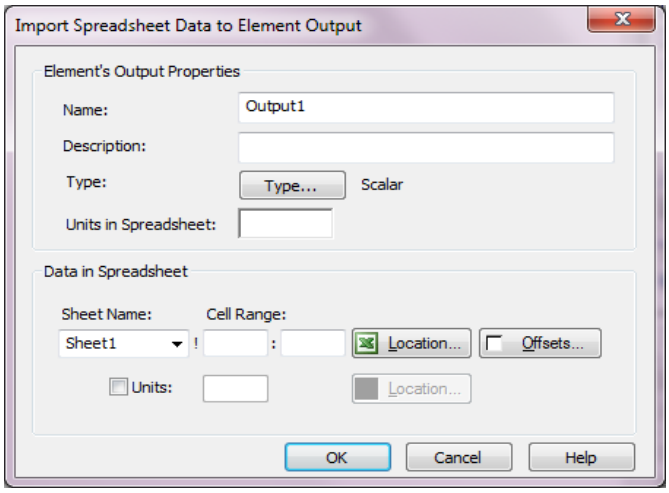
Distribution	Type Code	Distribution Parameters				
	Cell1	*Cell2*	*Cell3*	*Cell4*	*Cell5*	*Cell6*
Generalized Beta	BETAGEN	Mean	SD	Min	Max	
Beta	BETASF	Successes	Failures			
BetaPERT	BETAPERT	Min (0%) or 10%	Most Likely	Max (100%) or 90%	0 for 0/100; 1 for 10/90	
Binomial	BINOM	Batch Size	Prob			
Cumulative	CUM	# pairs	Linear = 0; Log = 1	Prob1	Val1	
Discrete	DISCRETE	# pairs	Prob1	Val1	Prob2	Val2
Exponential	EXPON	Mean				
Extreme Probability	EXTRPROB	Min = 0; Max =1	Number samples			
Extreme Value	EXTRVAL	Min = 0; Max =1	Location	Scale		
Gamma	GAMMA	Mean	SD	(Min)	(Max)	

Distribution	Type Code	Distribution Parameters				
	Cell1	*Cell2*	*Cell3*	*Cell4*	*Cell5*	*Cell6*
Log-Normal	LOGNORM	True = 0; Geom = 1	True or Geom Mean	True or Geom SD	(Min)	(Max)
Negative Binomial	NEGBINOM	Number successes	Prob of success			
Normal	NORMAL	Mean	SD	(Min)	(Max)	
Pareto	PARETO	Shape	Mode	(Max)		
Pearson Type III	PEARSON	Location	Scale	Shape		
Poisson	POISSON	Expected Value				
Sampled Results	SAMPLED	# Results	Val1	Val2	Val3	Val4
Student's t	STUDENTT	Deg of Freedom				
Triangular	TRIANG	Linear =0; Log =1	Min (0%) or 10%	Most Likely	Max (100%) or 90%	0 for 0/100; 1 for 10/90
Uniform	UNIFORM	Linear =0; Log =1	Min	Max		
Weibull	WEIBULL	Min	Slope	Mean – Min	(Max)	

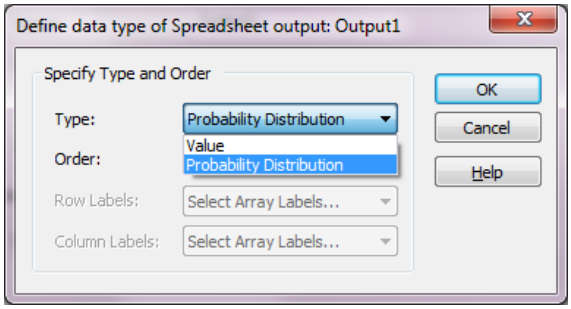
Table Notes:

- *Cumulative and Discrete can have as many Prob/Val pairs as required.*
- *Sampled Results can have as many Values as required.*
- *Parameters in parentheses are optional.*
- *If Log-Normal distribution is defined using Geometric parameters, the SD is dimensionless*
- *Shaded parameters are dimensionless; others have same dimension as the distribution*
- *There is no option to import a Boolean distribution.*

To import a distribution from a spreadsheet, you must press the **Type** button in the Import dialog:



After doing so, the following dialog will be displayed. You must select "Probability Distribution" as the Type:



The Order of a Probability Distribution must be Scalar.

When you specify the Location, you must select a single cell: that corresponding to the Type Code for the distribution. GoldSim will automatically read in the appropriate number of additional cells based on the distribution type.

Note: Prior to running the model, you can view any spreadsheet data that you plan to import by first selecting the **Update Spreadsheet Outputs** item in the menu displayed via the **Options>>** button for the Spreadsheet element, and then viewing the outputs of the element in tool-tips within the element's output port.

Note: Care must be taken when importing distributions that represent dates and temperatures. This is because dates and temperatures both have an absolute reference (i.e., 12/31/1899 for dates and absolute zero for temperatures). *Differences* between dates are expressed in units of time (e.g. days, seconds, etc) and *differences* between temperatures are expressed in 'deg' units (Cdeg, Fdeg). This can lead to errors when importing Stochastics, since GoldSim assumes a single unit for the distribution parameters, while some types of distributions would require two different types of units. For example, a Normal distribution representing a temperature would require the Mean to be specifed in absolute units (e.g., C) and the Standard Deviation to be specified in difference units (e.g., Cdeg). Similarly, a Normal distribution representing a date would require the Mean to be specifed in absolute units (e.g., as a date) and the Standard Deviation to be specified in difference units (e.g., days). If faced with this problem, there are several approaches for addressing this: 1) Use a distribution type where all parameters have the same units (e.g., triangular, uniform, cumulative); or 2) Specify and import the distributions as dimensionless and then apply a unit to the sampled value using an Expression element. This latter option would work for temperatures, but not dates, since a date cannot be specified with dimensionless values. It could, however, be imported as a Julian date, in which case you could use day units (or some other time unit). Alternatively, the mean date could be defined as a Data element and an imported distribution with time units could be used to model a 'lag' (positive or negative) from the fixed date. An Expression element could be used to add the sampled lag value to the fixed (mean) date.

Read more: Dealing with Temperature Units (page 97).

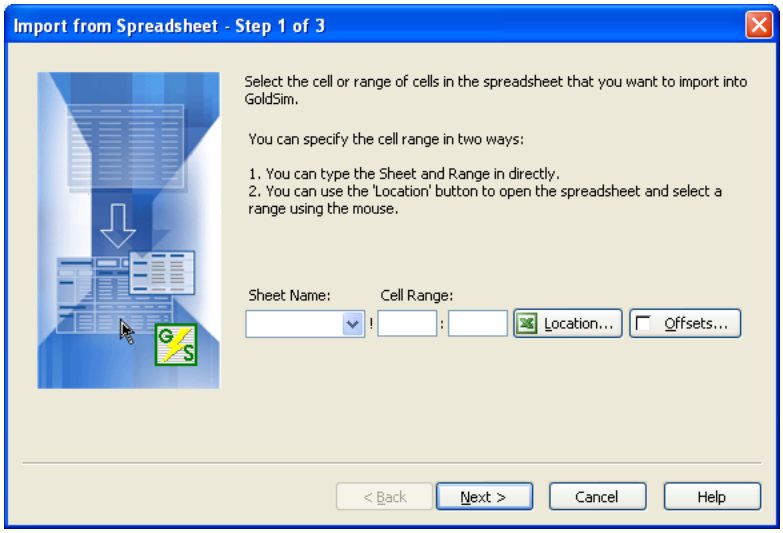
When a distribution is imported in this way, the output type on the spreadsheet element is not a value; rather, it is a complex output referred to as a Distribution output. Distribution outputs are complex outputs that represent all the statistical information necessary to define a probability distribution and only be used in several specialized locations.

To import this information into a Stochastic element, you must then use this Distribution output as the definition for a Stochastic element specified as an Externally-Defined distribution.

Read more: Externally-Defined Distribution (page 168).

Using the Spreadsheet Wizard to Define Spreadsheet Outputs

If you press the **Import…** button from the Import or Export Selection dialog, and the checkbox for using the wizard is checked, the Spreadsheet Wizard will be displayed:



This wizard is used to create an output to the Spreadsheet element. An output to a Spreadsheet element is defined by specifying a cell range in the spreadsheet file that you wish to import from the spreadsheet into GoldSim, making the data accessible within GoldSim as an element output. Until you become comfortable creating inputs and outputs for Spreadsheet elements, it is recommended that you use the wizard.

The wizard has three pages:

1. On the first page of the wizard, you select the location in the spreadsheet from which you wish to import the data. You can do this specifying the Sheet Name and Cell Range directly. Alternatively, you can press the **Location…** button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell range using your mouse.

Note: You cannot use the wizard to import a probability distribution from a spreadsheet. To do this, you must define the Import manually.

Read more: Importing Stochastic Element Definitions from a Spreadsheet (page 860).

2. On the second page, you must specify the **Units in Spreadsheet**. These represent the units of the data in the spreadsheet. This will also become the display units for the output after it is imported into GoldSim. If the range you have selected in Step 1 is more than a single cell (indicating that the data will be imported into GoldSim in the form of a vector or matrix), you will be prompted for the Array Labels for the vector or matrix.

Note: You cannot use the wizard to import the units for the data from a spreadsheet. To do this, you must define the Import manually.

Read more: Spreadsheet Element Outputs: Importing Data from the Spreadsheet (page 857).

3. On the last page of the wizard, you specify the **Name** of the output. It will default to Output*n*, where n is an integer. You will use this name to reference the output within GoldSim (as SpreadsheetElementName.OutputName). Hence, you will likely want to change this name from its default.

Note: If you specify a spreadsheet range that is more than one cell, the size of the range specified in Step 1 must be consistent with the size of the Array Label set(s) specified in Step 2. Hence, a vector output must map to a single column or row range with the number of rows or columns matching the number of items in the vector; and a matrix output must map to a rectangular range with the number of rows and columns matching the dimensions of the matrix (e.g., a matrix with 3 rows and 6 columns would have to map to a range in the spreadsheet 3 rows long and 6 columns wide).

If required, you can control the sheet and cell range into which GoldSim exports data dynamically. For example, you could instruct GoldSim to change the location to which it exports data based on the simulation time or the realization.

Read more: Defining Offsets for Inputs and Outputs to a Spreadsheet Element (page 865).

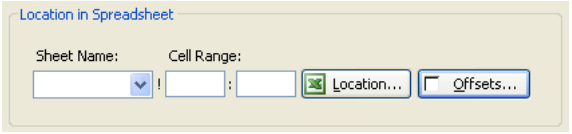
Note: Prior to running the model, you can view any spreadsheet data that you plan to import by first selecting the **Update Spreadsheet Outputs** item in the menu displayed via the **Options>>** button for the Spreadsheet element, and then viewing the outputs of the element in tool-tips within the element's output port.

Defining Offsets for Inputs and Outputs to a Spreadsheet Element

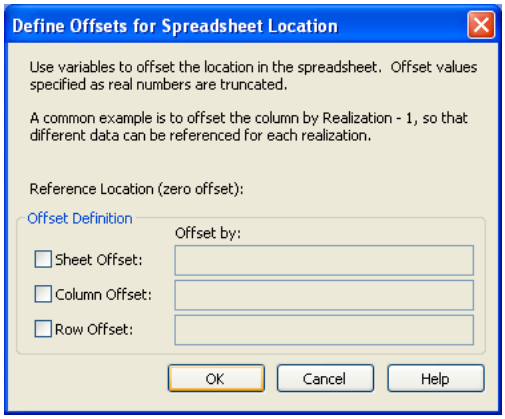
In some cases, you may want to specify the location in the spreadsheet file to which you are linking in terms of an offset from a defined location, in which the offset is defined by model variables (such as Time or Realization).

For example, you could instruct GoldSim to change the location to which it exports data based on the realization number, with results from each realization being exported to a different column in the spreadsheet file.

You do this by selecting the **Offset…** button when you are selecting a spreadsheet location while defining inputs or outputs to a Spreadsheet file:

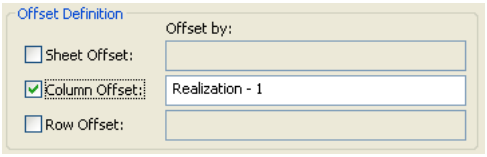


When you press this button, the following dialog is displayed:



You select whether you want to offset by Sheet, Column, and/or Row by clicking the appropriate checkboxes. For example, if you were exporting a scalar GoldSim output to the spreadsheet file, and wanted to export the value to Sheet1!A1 for realization 1, Sheet1!B1 for realization 2, and so on, you would simply do the following:

1. Specify Sheet1!A1 for the Location in Spreadsheet
2. Define the Column offset as follows:



In this case, the offset would be 0 for Realization 1 (and hence the value would be exported to Sheet1!A1), it would be 1 for Realization 2 (and hence the value would be exported to Sheet1!B1), and so on.

Note: The offsets can be any expression with links to other GoldSim outputs, but should be specified as dimensionless, integer values. GoldSim will not accept expressions in these fields that have dimensions. Offsets defined as real numbers are truncated.

Note: If as a result of an offset, you try to access a cell that is invalid (e.g., if you specify a negative offset from the first column), GoldSim will display a fatal error message at runtime.

If you wish to use a particular unit of time to define an offset, you must *cast* the output to a dimensionless number. For example, Time|day| will represent the number of days (which could be fractional, and hence will be truncated). Note that Run Properties like Day and Year are, by definition, dimensionless.

Read more: Unit Casting (page 98); Understanding and Referencing Run Properties (page 445).

When you close the Offset dialog and have a defined offset, this is indicated in the Spreadsheet Location (**Offset…** is checked) and in the list of inputs and outputs (any offset ranges are indicated as such).

Warning: In order to properly use offsets, you must have a good understanding of the conditions and settings that control when GoldSim exchanges data with the spreadsheet file.

Read more: Controlling When GoldSim Exchanges Data with the Spreadsheet File (page 869).

An example model which includes an illustration of the use of offsets for a Spreadsheet element (Spreadsheet.gsm) can be found in the General Examples folder in your GoldSim directory.

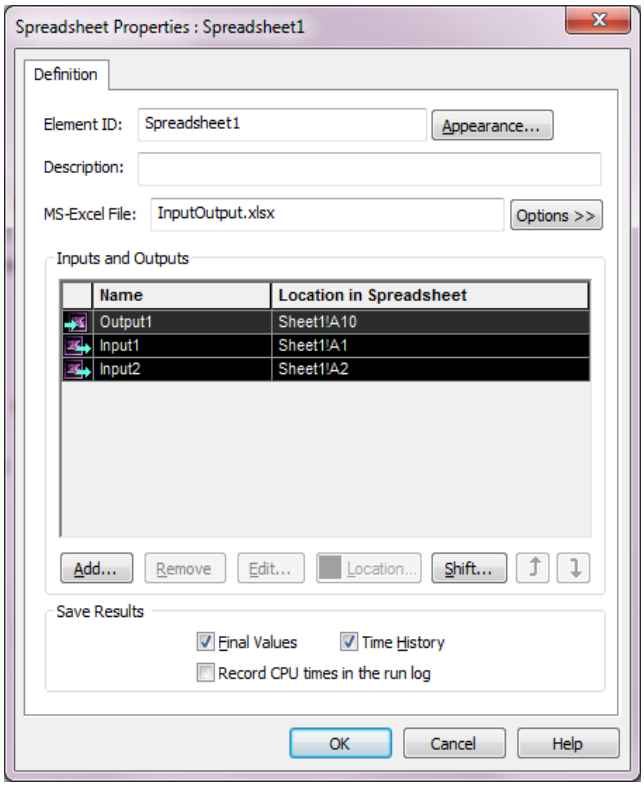
Shifting Ranges for Inputs and Outputs to a Spreadsheet Element

In some cases, you may want to edit the location of several inputs (or outputs) simultaneously by shifting them by sheet, row or column.

This could happen, for example, if you have multiple Spreadsheet elements linked to a single spreadsheet with multiple sheets. You may want each Spreadsheet element to link to a different sheet of the file, but otherwise, you would like all of the row and columns in the links to be identical.

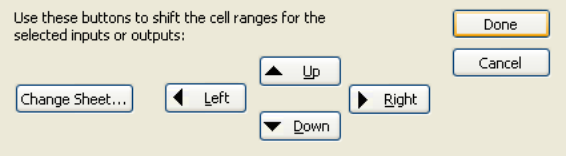
GoldSim provides a very convenient way to shift multiple inputs or outputs by sheet, row, and/or column.

You do this by selecting the items that you want to shift, and pressing the **Shift…** button from the main Spreadsheet element dialog:



Multiple contiguous rows in this dialog can be selected by holding down the **Shift** key while selecting with the mouse. Multiple rows that are not contiguous can be selected by holding down the **Ctrl** key while selecting with the mouse. All rows can be selected by placing the cursor in any row and pressing **Ctrl-A**.

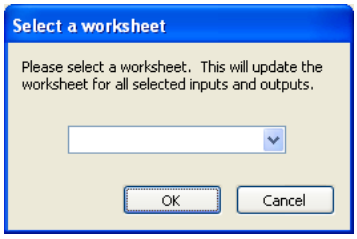
When you select items and press the **Shift…** button, the Spreadsheet element dialog expands to show the following section:



The **Up** and **Down** keys shift the selected items by row. The **Left** and **Right** buttons shift the selected items by column.

Note: If you try to shift beyond the valid range for any of the selected items (e.g., if you try to shift left and one of the items is defined as being in column A) GoldSim will not shift any of the items.

The **Change Sheet…** button displays a dialog for selecting a new sheet to apply to all selected items:



The drop-list includes all worksheets present in the Excel file. By default, no new sheet is selected. If you press OK without selecting a sheet, no changes are made. If you select a sheet and press OK, all selected items will be updated (they will point to that sheet).

Note: If you are importing both the data and the units for the data from a spreadsheet, when you shift the data, GoldSim will also shift the reference to the unit if a) you are shifting rows, and the data and the units are in the same row; b) you are shifting columns, and the data and the units are in the same column; or c) you are shifting sheets (by defintion, the units are always in the same sheet as the data).

Locking onto a Spreadsheet File

GoldSim allows you to "lock onto" the spreadsheet file that is being referenced. When you lock onto a file (by selecting the **Lock onto selected file** item in the menu displayed via the **Options>>** button), the following additional information regarding the referenced file is saved with the element:

- File and path name;
- Date the file was created;
- Date the file was last modified;
- File size; and
- CRC signature.

Once you have locked onto a file, the CRC signature is displayed in a tool-tip when you hold your cursor over the filename in the dialog.

The CRC signature is an alphanumeric code that can be used to uniquely identify whether the file contents have changed. When you run a model that has locked onto a file, GoldSim compares the CRC signature of the file with the

original signature that was stored. If these are not identical (indicating that the file has been changed), GoldSim displays an error message and will not run the model.

You can unlock a file by clearing the **Lock onto selected file** item. Note that if versioning is enabled, whenever a file is locked onto or unlocked, this information is logged with the version.

Read more: Tracking Model Changes (page 963).

Note: Saving changes to a spreadsheet file and file locking are mutually exclusive. If the **Save MS-Excel file after simulation** item is selected in the **Options>>** menu when you try to lock onto a file, GoldSim will warn you that locking onto the file will automatically disable saving changes to the MS-Excel file. When you lock onto a file, the **Save MS-Excel file after simulation** item is no longer available in the **Options>>** menu.

Note: The act of opening a workbook in Excel automatically modifies the file, even if no changes are made. In order to allow spreadsheet files to be viewed without changing the CRC signature, GoldSim automatically makes the file read-only when it is locked onto. Note that if it is subsequently unlocked, it remains read only, and you will need to change this manually if you wish to edit the file.

Read more: How Spreadsheet Files are Affected by GoldSim (page 871).

Controlling When GoldSim Exchanges Data with the Spreadsheet File

Whenever a Spreadsheet element is used in a model, GoldSim uses the following information to determine when during the simulation it should import data from or export data to the spreadsheet file:

- Is the **Recalculate in Excel during simulation** setting (accessed via the **Options>>** button) On or Off? This defaults to being On (yes) when a new Spreadsheet element is created.
- Are you importing data from the spreadsheet, exporting data to the spreadsheet, or both?
- Are offsets defined, and if so, has an offset value changed?
- If you are exporting data to the spreadsheet, have any of the GoldSim outputs being exported changed?

The following table summarizes how this information is used to determine when to exchange data with the spreadsheet file:

Type of Exchange Specified in Element	Recalculate Setting	Offsets defined	When does GoldSim exchange data with spreadsheet file?
Export only	Not available	No	Export all data at end of simulation.
	Not available	Yes	Export data from just before its offset changes.* Export all data at end of simulation.

Type of Exchange Specified in Element	Recalculate Setting	Offsets defined	When does GoldSim exchange data with spreadsheet file?
Import only	Not available	No	Import all data at beginning of simulation.
	Not available	Yes	Import all data at beginning of simulation. Import data when its offset changes
Export and Import	Off	No	Export all data at end of simulation. Import all data at beginning of simulation.
	On	No	Export and Import all data at beginning of simulation. Export and Import all data whenever an Export variable changes. Export all data at end of simulation.
	Off	Yes	Export and Import all data at beginning of simulation. Export data from just before its offset changes.* Export all data at end of simulation. Import data when its offset changes
	On	Yes	Export and Import all data at beginning of simulation. Export and Import all data whenever an Export variable changes. Export data from just before its offset changes.* Export all data at end of simulation. Import data when its offset changes

**The value that is exported when an export offset changes is the value from the update immediately before the offset changed. Hence, if the offset was "Realization", when Realization changes from 1 to 2, GoldSim would export the value from the end of Realization 1 using an offset of 1. Similarly, if the offset was "Month", when Month changes from 1 to 2, GoldSim would export the value from the end of Month 1 using an offset of 1.*

Several points about spreadsheet recalculations are worth noting:

- Regardless of the **Recalculate in Excel during simulation** setting, by default, Excel will recalculate whenever cell values change. As pointed out below, however, GoldSim will only retrieve the updated data if **Recalculate in Excel during simulation** is selected. You can change this setting in Excel (so it does not automatically recalculate) in the Calculation tab of Excel's Options dialog.
- The **Recalculate in Excel during simulation** setting does two things: 1) it forces Excel to recalculate, and 2) it retrieves the recalculated data back into GoldSim. Hence, if the **Recalculate in Excel during simulation** setting is off, GoldSim will *not retrieve updated data* from the spreadsheet, even if some cells have changed (and Excel automatically recalculated).

Note: When you use a Spreadsheet element, there may be a delay of several seconds at the start of a simulation as Excel is loaded into memory. Once it is loaded, however, the simulation should progress rapidly.

Note: If you are only importing data from a spreadsheet *and* you have not specified any dynamic offsets *and* you have already imported the data previously, if GoldSim cannot find the spreadsheet when you try to run the model it will use the data that was imported previously and will log a warning message to the Run Log.

Warning: You should never try to open a spreadsheet that is linked to GoldSim while a simulation is running. In addition, prior to running a GoldSim model that is linked to a spreadsheet, all referenced spreadsheets should be closed. If you try to interact with a spreadsheet that is being used by GoldSim during a simulation, all interaction with the spreadsheet is blocked by GoldSim.

If you are importing data from a spreadsheet, you can tell GoldSim to import the data while in Edit Mode by selecting **Update Spreadsheet Outputs** via the **Options>>** menu button. When you do this, GoldSim will import the data currently existing in the spreadsheet. If the information being imported is impacted by data that is being exported to the spreadsheet (and that data can be computed in Edit Mode), GoldSim will export the data to the spreadsheet, trigger a recalculation, and import the data from the spreadsheet to GoldSim, where it can be viewed as an output of the spreadsheet element.

How Spreadsheet Files are Affected By GoldSim

By default, GoldSim's interactions with the Excel spreadsheet only affect a copy of the spreadsheet temporarily stored in memory, and do not affect the data permanently stored in your spreadsheet file. That is, following your simulation, the spreadsheet file will be unchanged (even if during the simulation GoldSim has sent data to cells in the spreadsheet).

In some cases, however, you may want to save the changes you make to a spreadsheet (i.e., when GoldSim exports data to the spreadsheet). To do so, select the **Save MS-Excel file after simulation** item in the **Options>>** menu (which by default is cleared).

Note: Unless you use offsets that vary with time or realization, if you access the spreadsheet multiple times (at multiple timesteps or for multiple realizations), only the changes made to the spreadsheet the final time you access it will be saved (all the previous modifications will be overwritten each time that you access it).

Read more: Defining Offsets for Inputs and Outputs to a Spreadsheet Element (page 865).

Note: Saving changes to a spreadsheet file and file locking are mutually exclusive. If the **Save MS-Excel file after simulation** item is selected in the **Options>>** menu when you try to lock onto a file, GoldSim will warn you that locking onto the file will automatically disable saving changes to the MS-Excel file. When you lock onto a file, the **Save MS-Excel file after simulation** item is no longer available in the **Options>>** menu.

Read more: Locking onto a Spreadsheet File (page 868).

Warning: You should always close any spreadsheets that are referenced by GoldSim prior to running your GoldSim model. In fact, in most cases, GoldSim will warn you if it tries to interact with (in particular, write to) a spreadsheet that is already open.

Exchanging Date Information with a Spreadsheet

In some cases, you may wish to exchange date or time information with a spreadsheet. For example, you may wish to send the current date in the GoldSim simulation to a cell formatted as a date in a spreadsheet, or retrieve a date from a spreadsheet into GoldSim. In order to do so, you must understand how Microsoft Excel (and GoldSim) deal with dates.

Within Excel, a date is stored internally as the number of days (which can be fractional) since December 31, 1899 00:00:00. Hence, if you were to export the number 4.25 into a spreadsheet cell formatted as a date, it would be interpreted as January 4, 1900 at 6:00:00 AM. However, Excel mistakenly adds an extra day into its calendar that did not actually exist (February 29, 1900). As a result, the *effective* Julian date reference for all times after March 1, 1900 in Excel is actually December 30, 1899 00:00:00.

Within GoldSim, the Run Property DateTime represents the elapsed time since the reference date December 30, 1899 00:00:00. Hence, for all dates after March 1, 1900, Excel and GoldSim have the same effective Julian date reference.

Given this information, the following rules should be followed for exchanging date information between GoldSim and a spreadsheet:

- When exporting a date/time from GoldSim to a spreadsheet cell, you should select Days for the **Units in Spreadsheet**. If the spreadsheet cell is formatted as a date, it will display as a date; otherwise, it will display the number of days from December 30, 1899 00:00:00 to the specified date that was exported.
- When importing a date from a spreadsheet cell (i.e., a cell formatted in the spreadsheet as a date), you should select Date or Datetime for the **Units in Spreadsheet**. Within GoldSim, the output from the Spreadsheet element would then be stored as the number of days from December 30, 1899 00:00:00 to the particular date that was imported from the spreadsheet. If running a Calendar Time simulation, the output would be displayed as a date when viewing it (e.g., using a tool-tip) in GoldSim.

Read more: Referencing Dates in Expressions (page 135); Setting the Basic Time Options (page 413).

Saving Spreadsheet Element Outputs

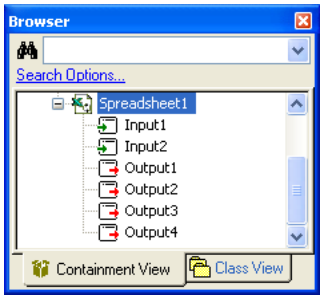
You can save results for a Spreadsheet element by clicking **Final Values** and/or **Time Histories**. Checking one of these causes all output arguments to be saved.

If the element has multiple outputs, and you wish to save only one or two of these as results, you can use the context menu of the output in the browser (accessed via a right-click on an output) to turn on or off a particular output. In this case, the checkbox in the dialog would then become a box instead of a check (indicating that only some of the results will be saved). Note that in order to see outputs in the browser you must first ensure that **Show element subitems** is selected by right-clicking anywhere in the browser.

Note: A Spreadsheet element does not have a primary output.

Viewing a Spreadsheet Element in the Browser

The browser view of a Spreadsheet element is shown below:



As can be seen, the browser view shows an item for each specified input to the spreadsheet, and an item for each specified output.

Note: Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

Read more: Using the Browser (page 110).

External (DLL) Elements

In some situations, you may wish to define a complex function which can not be readily implemented using the expression editing features supplied by GoldSim. For example, calculation of an output may require very complex logic which would be cumbersome to represent using a Selector element, or it may require a numerical solution technique (e.g., iteration).

GoldSim provides two separate ways to deal with such situations:

- You can develop separate program modules (written in C, C++, FORTRAN or other compatible programming languages) which can then be directly coupled with the main GoldSim algorithms. These user-defined modules are referred to here as external functions, and the elements through which they are coupled to GoldSim are called External (DLL) elements (page **Error! Bookmark not defined.**).
- You can specify a sequence of statements directly within the GoldSim interface using a Script element.

Read more: Script Elements (page 803).

In general, using Script elements is prefered, as this approach has the key advantages that 1) it does not require use of a separate programming language and interface; and 2) it is much more transparent (all of the "code" can be seen directly in GoldSim). However, in some cases (e.g., linking complex existing

programs directly into GoldSim), Script elements cannot be used and External (DLL) elements are the only option.

The modules are linked into GoldSim as DLLs (Dynamic Link Libraries) at run time. Integrating these external modules into GoldSim requires that you develop a "wrapper" (or "shell") around your existing function, and compile it into a DLL. In most cases, this will require only a limited number of programming modifications. GoldSim supports both 32-bit and 64-bit DLLs.

An example model which uses an External (DLL) element (External.gsm) can be found in the General Examples/External folder in your GoldSim directory. This folder also includes the corresponding DLL, as well as the source code (in C++ and Fortran) for the example.

Implementing the External Module

The steps involved in creating and using an External element are as follows:

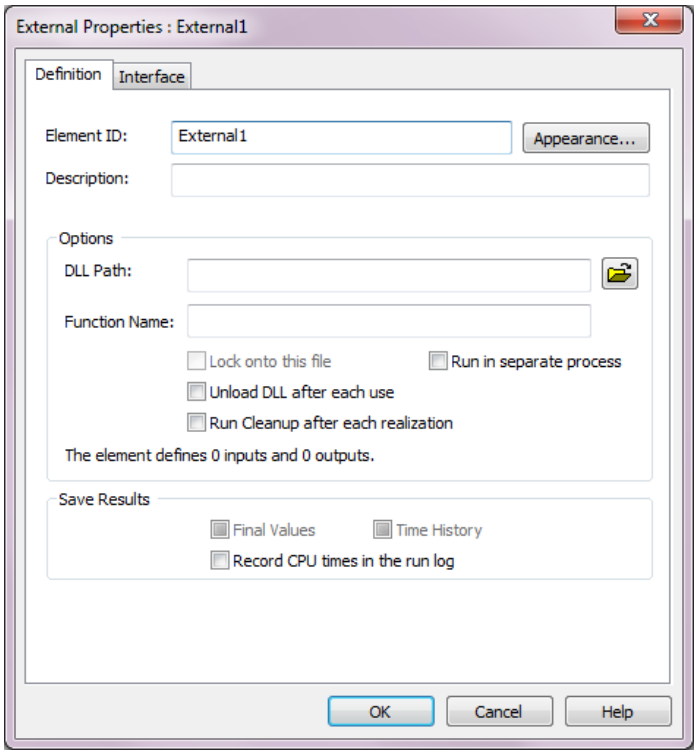
1. Code the external function to be used by the External element in your model;
2. Compile the external function into a DLL (multiple functions can be included in the same DLL);
3. Create an External element within GoldSim and specify the DLL and the specific function within the DLL to be used;
4. Specify the input and output arguments that are to be sent to and received from the function.

The details of the first two steps are discussed in Appendix C. The remaining two steps are discussed below.

Read more: Appendix C: Implementing External (DLL) Elements (page 1033).

Defining the External Element

The editing dialog for an External element is shown below:



In order to use an External element, you must first specify the **DLL Path** and **Function Name** within the DLL that the element will utilize:

DLL Path: This is the name of the dynamic link library containing the external function. You can press the Browse button (the folder to the right of the **DLL Path** field) to search for the file. In general, you should specify the full path to the DLL. (If you specify just the filename, it will look for the DLL only in the working directory containing the GoldSim model file). Note that a single DLL can contain multiple external functions.

Note: If the DLL references any external files and it is run in a separate process, it will look for those files in the folder where the DLL resides. If it is not run in a separate process, it will look for the files in the folder where the model file resides.

Function Name: This is the specific name of the external function in the DLL. This name is case-sensitive and must exactly match the name of the function in the source code for the external function.

There are also several options that control how the DLL is called by GoldSim:

Unload DLL after each use: If this box is checked, GoldSim will unload the DLL (and continue the simulation). This is useful when running very large model files (in which the DLL only needs to be called infrequently). If the DLL is subsequently called again, GoldSim will automatically reload it.

Run Cleanup after each realization: If this box is checked, GoldSim will call the DLL with a cleanup instruction (99) at the end of each realization.

Read more: Appendix C: Implementing External (DLL) Elements (page 1033).

Lock onto this file: If this box is checked, GoldSim regards various properties of the file to determine whether the file contents have changed.

Read more: Locking onto a DLL File (page 879).

Run in separate process: If this box is checked, GoldSim executes the DLL outside of the GoldSim process space. This can be useful for DLLs that have large memory requirements.

Read more: Running the DLL in a Separate Process (page 879).

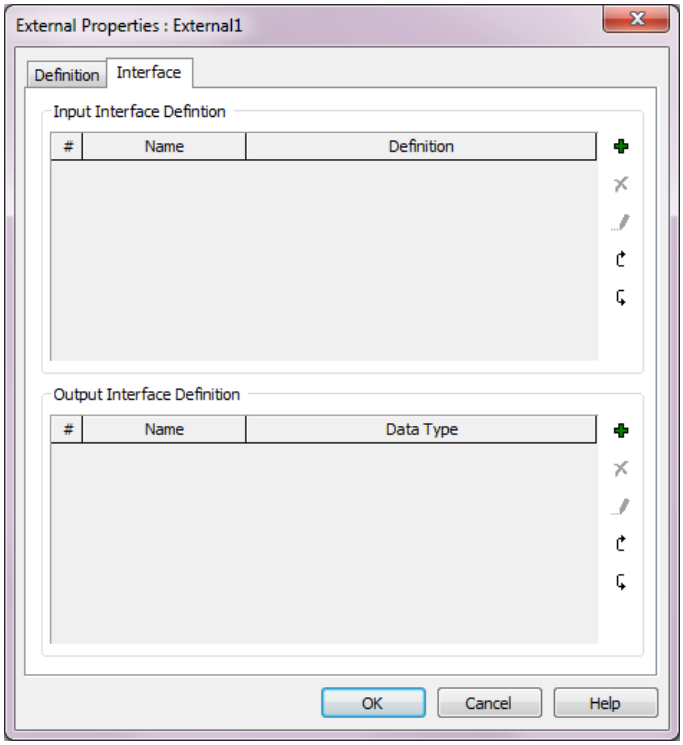
Note: You can use GoldSim's File element to automatically copy the DLL file from some location (typically on a network) to your machine. Because GoldSim records this action in the Run Log, it can provide an "audit trail" to ensure that the proper version of an external file (such as a DLL) was used in a particular simulation.

Read more: File Elements (page 884).

If you check the **Record CPU times in the run log** box, if the element uses more than 1 CPU seconds, a message will be written to the GoldSim run log identifying the element's name, type (i.e., External), the number of times it was updated, and the total CPU time used.

Read more: The Run Log (page 506).

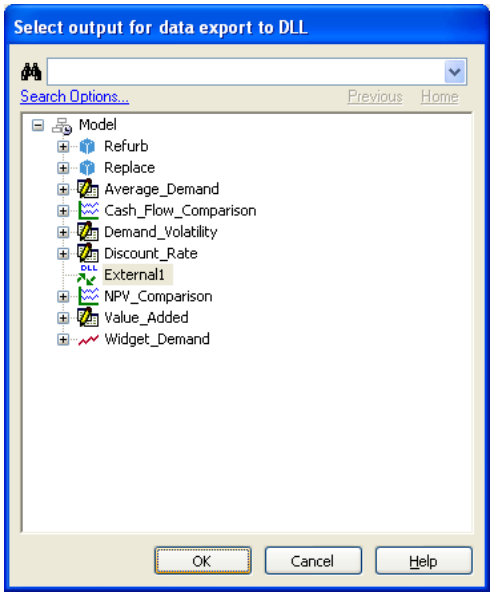
You define the input and output arguments for the External element via the **Interface** tab:



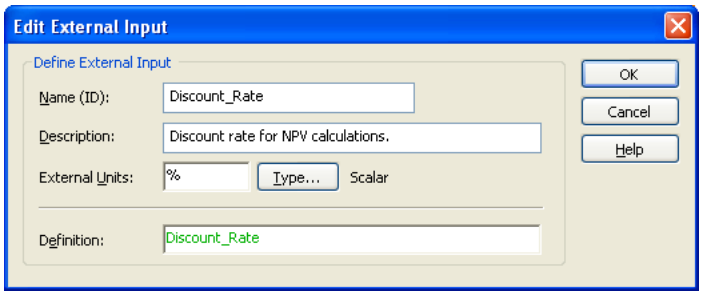
The top part of this dialog is used to define the input arguments. The bottom portion is used to define the output arguments.

Add button

In order to add an input argument, press the Add button in the Input Interface portion of the dialog. When you do this, GoldSim displays the Insert Link dialog, allowing you to select an existing output to add to the interface):



After selecting a link and pressing **OK**, the following dialog is displayed:



The **Definition** displays the output that was selected. The **Name (ID)** defaults to the output name. The **Name (ID)** will appear on the input interface for the External element (accessed via the input port). The **Description** is optional and simply provides additional information regarding the input argument (it defaults to the element description for the output that was selected).

The dimensions of the **External Units** and **Type…** (e.g., value/condition, scalar/vector/matrix) for the input argument are automatically determined by the output that was selected. Prior to sending the argument to the DLL, GoldSim converts it to a number in the **External Units** specified. This can be edited, but must have dimensions consistent with the **Definition**.

Note: If you send an output whose **Type** is a Condition to an external function, GoldSim will automatically convert it to either a 0 (for False) or 1 (for True) prior to sending the argument to the DLL.

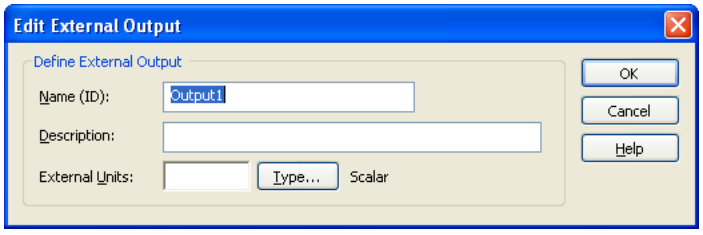
Note: If you pressed **Cancel** (rather than selecting an output) when adding the input argument, the **Definition** (and hence **External Units** and **Type**) will be blank. You could enter any expression into the **Definition** field (including constants). The **External Units** and **Type** would need to be specified in a manner that was consistent with the **Definition**.

Note: In addition to reading in Values or Conditions, an External function can also read in (and subsequently output) a Time Series Definition.

Read more: Using an External Element to Read and/or Output Time Series (page 881).

Add button

In order to add an output argument, press the Add button in the Output Interface portion of the dialog. When you do this, GoldSim displays the following dialog:



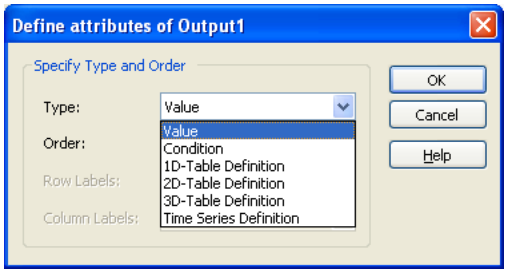
The **Name (ID)** is the name by which the output variable can be referenced in the model. For example, if the name of the External element was X, and the

name of the output was Y, the output could be referenced as X.Y. This ID has the same restrictions as an element ID.

The **Description** is an optional description of the output. The **External Units** represent the units in which the External function returns the value. The specified units also become the display units for the output. The **Type...** button is used to access a dialog for specifying the data type (e.g., value/condition, scalar/vector/matrix).

Note: If you add an output whose **Type** is a Condition to the output interface, GoldSim will treat any non-zero number as True, and zero as False.

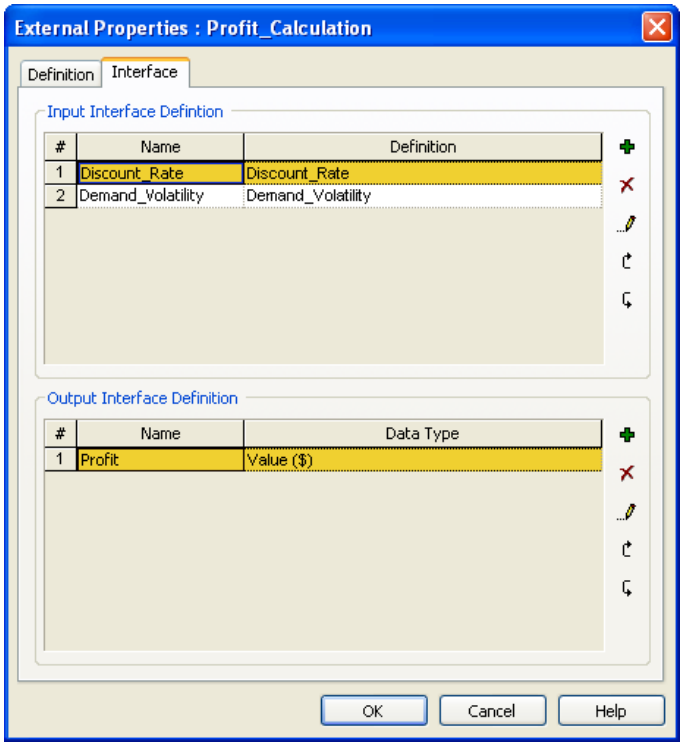
In addition to defining the output as a Value or a Condition, the output can also represent either a Lookup Table or a Time Series:



Read more: Using an External Element to Define Lookup Tables (page 880); Using an External Element to Read and/or Output Time Series (page 881).

There is no limit to the number of input and output arguments that can be added.

After adding the input and output arguments, the Interface tab displays them as follows:



The input arguments and output arguments are transferred between GoldSim and the external function in exactly the order in which they are listed on the **Interface** tab.

Buttons for deleting, editing and moving the input and output arguments up and down in the list are available directly below the Add buttons.

Locking onto a DLL File

GoldSim allows you to "lock onto" the DLL file that is being referenced. When you lock onto a file (by checking the **Lock onto this file** option in the External element dialog), the following additional information regarding the referenced file is saved with the element:

- File and path name;
- Date the file was created;
- Date the file was last modified;
- File size; and
- CRC signature.

Once you have locked onto a file, this information is displayed in a tool-tip when you hold your cursor over the filename in the dialog.

The CRC signature is an alphanumeric code that can be used to uniquely identify whether the file contents have changed. When you run a model that has locked onto a file, GoldSim compares the CRC signature of the file with the original signature that was stored. If these are not identical (indicating that the file has been changed), GoldSim displays an error message and will not run the model.

You can unlock a file by clearing the **Lock onto this file** checkbox. Note that if versioning is enabled, whenever a file is locked onto or unlocked, this information is logged with the version.

Read more: Tracking Model Changes (page 963).

Running the DLL in a Separate Process

By default, GoldSim loads DLLs used by External elements into the GoldSim process space. GoldSim and the all external DLLs therefore share the application's address space, which is usually limited to 2 GB on Windows operating systems. In some case, a single DLL itself may have large memory requirements (e.g., if it loaded large sets of data from files or databases), or it could statically or dynamically allocate large blocks of memory. If the combined memory requirements of GoldSim (the executable), the GoldSim model (including all model properties, buffers and result values) and all DLLs exceeds 2 GB, GoldSim will be unable to carry out the simulation. The user will be forced to preserve memory by saving fewer histories or time steps, or by turning off saving of results for certain elements.

To circumvent this limitation, GoldSim provides a feature that allows external DLLs to be executed outside of the GoldSim process space. If **Run in separate process** is checked, GoldSim does not load the DLL into its process space. Instead, it launches a separate small DLL server with its own private process space. The DLL server works like a middleman who dispatches information between GoldSim and the external DLL. Each DLL that runs in a separate process has its own DLL server. Therefore each DLL has access to a private 2 GB process space, which is not shared with any other DLL or application.

Note: If the DLL references any external files and it is run in a separate process, it will look for those files in the folder where the DLL resides. If it is not run in a separate process, it will look for the files in the folder where the model file resides.

Note: 64-bit DLLs *must* be run in a separate process.

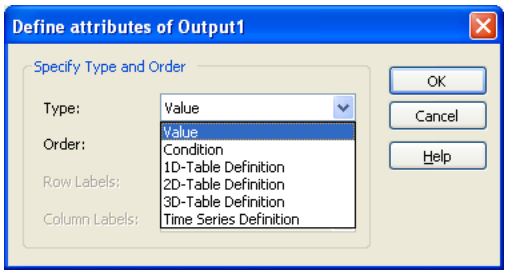
Using an External Element to Define Lookup Tables

In some cases, you may wish an external program to directly generate a look-up table that you can use to dynamically define a Lookup Table element.

Read more: Lookup Table Elements (page 263).

To facilitate this, GoldSim allows you to define the table using an External element. The entire table then becomes an output of the External element, which can subsequently be linked to a Lookup Table element.

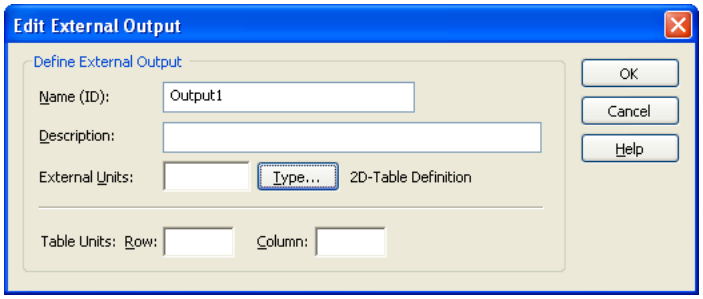
In order to specify that an output of the External element is a table, you must indicate that the Data Type of the output is a 1D, 2D, or 3D Table Definition (when you define the **Type…** for an output argument):



As seen above, in addition to defining an output of an External element as a Value or a Condition, you can also define an output whose data type is a 1D, 2D, or 3D Table Definition. These three special data types can only be produced by an External element and outputs of these types can only be used in special locations within GoldSim (i.e., an input defining a Table element's data).

Since the table itself is being defined by an external function (and will subsequently be linked to a Lookup Table element), it is necessary to specify the units for the numbers representing the table that are being passed into GoldSim by the external function.

As a result, when you add an output argument and define it as a Lookup Table, the dialog for defining the output differs depending on whether the data type is a 1D, 2D, or 3D Table Definition. In all cases, the **External Units** represent the units for the independent variable. For a 1D Table Definition, you must also define the units for a single independent (**Row**) variable. For a 2D Table Definition, you must define the units for the two independent (**Row** and **Column**) variables. For a 3D Table Definition, you must define the units for the three independent (**Row**, **Column** and **Layer**) variables. Here is an example of what the output argument dialog looks like for a 2D Table Definition:



You link a Lookup Table to an external function by selecting "External DLL" from the **Data Source** field at the bottom of the Lookup Table dialog:

When you do so, a new tab (**External DLL**) is added to the dialog. You use this tab to specify the name of the External element output that defines the table.

Note: In order to link a 1D, 2D, or 3D Table Definition output from an External element to a Table element, the dimensions of the External Units specified in the dialog for the output must be consistent with the Result Units and the independent variable units specified when defining the Lookup Table element.

Note: After you have defined a Lookup Table element externally and run the model (so that the DLL has loaded the table into the Lookup Table element), you can view the table definition created by the external function by pressing **View Data…**. In addition, if you have run the model and you subsequently "uncouple" the External element from the Lookup Table element (by selecting "None" from the **Data Source** drop-list in the Lookup Table dialog), the data in the table will become accessible (by pressing **Edit Table…**).

The specific format in which an external function passes table definitions to GoldSim is discussed in Appendix C.

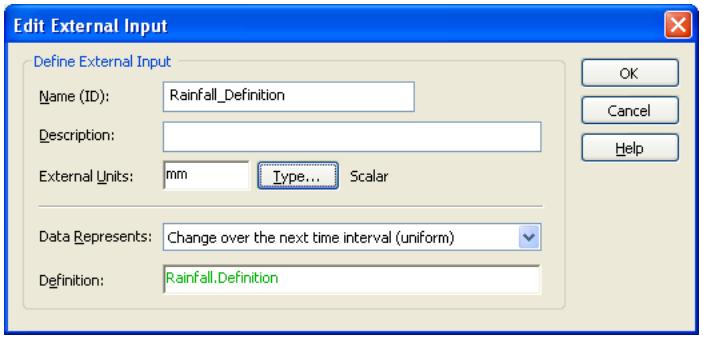
Using an External Element to Read and/or Output Time Series

In some cases, you may wish an external program to directly read, manipulate, and/or generate a Time Series Definition. A Time Series Definition output produced by such an external function can be used to dynamically define a Time Series element.

Read more: Referencing a Time Series Definition Output (page 226).

To facilitate this, GoldSim allows you to input and/or output a Time Series Definition using an External element.

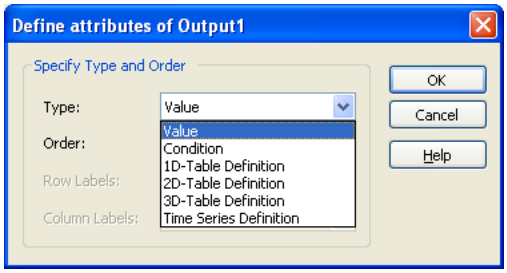
In order to specify that an *input* argument of the External element is a Time Series Definition, you must select a Time Series Definition output when you select the output to link to. After doing so, the input argument dialog will look like this:



Note that in addition to listing the **Definition** and **External Units**, it is also necessary to specify what the time series **Data Represents**. The **External Units** and **Data Represents** field must be consistent with the actual Time Series Definition that is referenced (and automatically default to the correct values when the link is created).

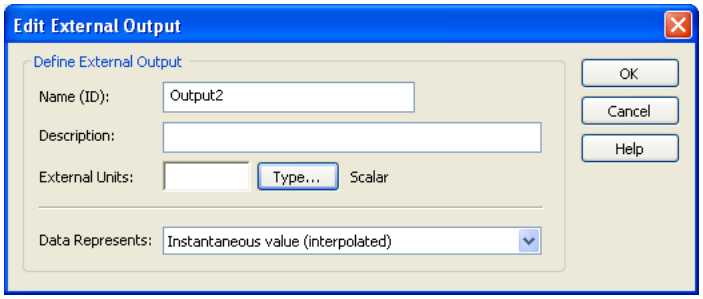
Read more: Specifying What the Input to a Time Series Represents (page 191).

In order to specify that an *output* argument of the External element is a Time Series Definition, you must indicate that the Data Type of the output is a Time Series Definition (when you define the **Type…** for an output argument):



As seen above, in addition to defining an output of an External element as a Value, Condition or Table, you can also define an output whose data type is a Time Series Definition. Time History Definition outputs are complex outputs that represent all the information necessary to define a time series.

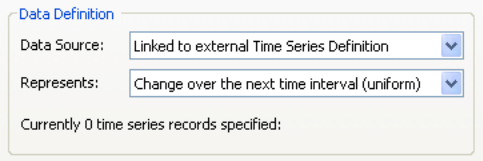
If you select Time Series Definition, the following dialog is displayed for specifying the attributes of the output:



Note that in addition to defining the **External Units**, you must also specify what the time series **Data Represents**.

Read more: Specifying What the Input to a Time Series Represents (page 191).

You link a Time Series Definition produced by an External element to a Time Series element by selecting "Linked to external Time Series Definition**"** from the **Data Source** field in the Data Definition section of the Time Series element dialog:



When you do so, a new tab (**Linked**) is added to the dialog. You use this tab to specify the name of the External element output that defines the time series.

Note: If using an External element to output a Time Series Definition, the DLL can only be called at etime = 0. Because a Time Series must be defined at the beginning of the simulation, if you try to redefine the time series in the middle of the simulation, GoldSim will issue a Fatal Error.

The specific format in which an external function reads and outputs Time Series Definitions is discussed in Appendix C.

When is the External Element Called?

The external function is called at the following times:

- At the beginning of the simulation (Etime = 0);
- Every GoldSim timestep (if one of the inputs to the element has changed); and
- At any internal step in which an Event is triggered (if one of the inputs to the element has changed).

Read more: How GoldSim Inserts Events into a Simulation (page 380).

It is likely that you may want to treat each of these calls in a different way. For example, you may wish the external function to ignore the call at the beginning of the simulation, or you may want it to carry out some special initialization routines at that time.

You may also want to treat a call in the middle of a timestep (due to an Event) differently from a call at the end of a timestep.

In order to communicate this kind of information to the external function, you would need to send Etime (the current elapsed time) and perhaps Timestep_Length (the length of the current timestep) to the external function as optional input arguments. (Both of these are Run Properties).

Read more: Understanding and Referencing Run Properties (page 445).

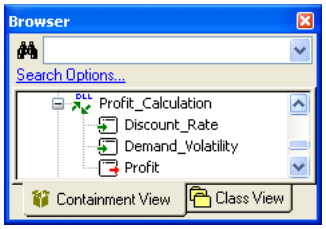
Saving External Element Outputs

You can save results for an External element by clicking Final Values and/or Time Histories. Checking one of these causes all output arguments to be saved. If the element has multiple outputs, and you wish to save only one or two of these as results, you can use the context menu of the output (accessed via a right-click) in the browser to turn a particular output on or off. In this case, the checkbox in the dialog will become gray (indicating that some of the results will be saved). The display units for these outputs are defined when the output is created.

Note: An External element does not have a primary output.

Viewing an External Element in the Browser

The browser view of an External element is shown below:



Note: Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

The browser view shows an item for each specified input argument, and an item for each specified output argument. The names of the inputs and outputs are determined by the **Name (ID)** specified when you added the argument to the interface.

File Elements

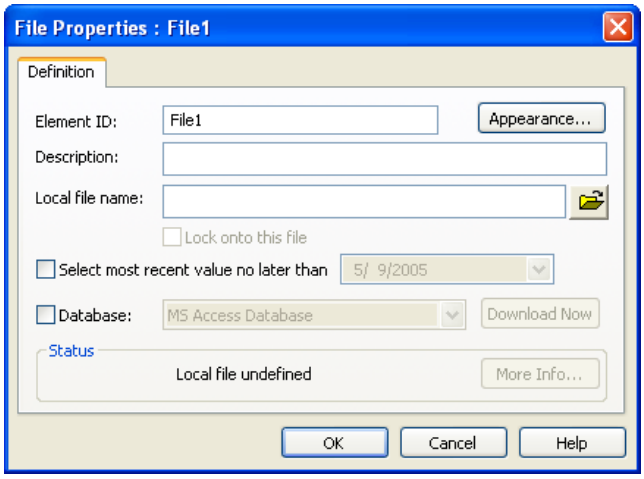
The File element is a support element in GoldSim. It has no inputs or outputs and carries out no calculations. Its purpose is to simply control copies of auxiliary files that may be required by External elements and Spreadsheet elements.

File elements have two uses:

- To ensure that all necessary support files are passed to Slaves when using the GoldSim Distributed Processing Module; and
- To ensure that support files which are stored on a network are accessed for use in a simulation, and to provide an "audit trail" of the file transfer.

In the latter case, the File Element utilizes a specifically formatted database to identify a "source" file, typically on a network, and then copy the target file onto the local computer. This file is then available for access by GoldSim elements.

The dialog for a File element looks like this:



Using File Elements to Support Distributed Processing

The most common use of a File element is to ensure all necessary support files are passed to Slaves when using the GoldSim Distributed Processing Module.

The Distributed Processing Module allows you to use multiple computers connected over a network to share the computational burden of a Monte Carlo

simulation. This is accomplished by having a Master (server) GoldSim executable connect to multiple Slave (client) GoldSim executables.

The first step in such a simulation is for the Master to transfer the necessary files (the model files and any support files) to the Slaves.

If the model uses any External or Spreadsheet elements, the Master automatically transfers the DLLs and spreadsheet file, respectively, to the Slaves. If, however, these elements reference any other files (e.g., if the DLL reads in a separate data file the first time it is called), GoldSim has no way of knowing this, and could not send the required files to the Slaves.

The File element can be used to overcome this problem. Any file referenced by a File element is automatically sent to Slaves in a distributed processing simulation.

Hence, when using the Distributed Processing Module with external function elements that reference other files, a File element should be created for each file.

The file name should be entered in the **Local file name** field of the File element, and the **Database** checkbox should be cleared. If no path is provided, the file should be placed in the same directory as the model file, and if the file cannot be found locally, the Status area displays "Local file missing", and the simulation is not allowed to proceed.

Note: The Distributed Processing Module is discussed in detail in the **GoldSim Distributed Processing Module User's Guide**.

Using File Elements to Access a File on a Network

A File element can be used to ensure that support files which are stored on a network are accessed for use in a simulation, and to provide an "audit trail" of the file transfer.

When used in this way, the **Element ID** for the File element is used as a key to access a record in a specifically formatted database referred to as a Yucca Mountain database. This record provides a network path-name for the source file, and also provides an integrity-checking code for the file (to ensure that the file was copied with no errors).

Read more: Downloading from a Yucca Mountain Database (page 978).

To use a File element in this way, the **Database** box must be checked, and you must enter the data source name associated with a Yucca Mountain database. Before using GoldSim's database features you must first define all available databases on your computer using Control Panel's 32-bit ODBC Data Sources option. This allows you to define a name for each data source, and link it to a specific database file.

Read more: Adding Data Sources to Your Computer (page 973).

When the **Download Now** button is pressed, or when a global database download is carried out, GoldSim locates the source file and makes a copy of it. The **Local file name** input field defines the local destination for the copy of the source file. If the local file already exists, it is automatically overwritten.

After copying, the CRC signature of the downloaded file is compared to that of the file entered in the database to ensure the integrity of the data transfer. (The CRC signature is an alphanumeric code that can be used to uniquely identify whether the file's contents have changed.) If the codes are different, the download is treated as having failed.

As is the case for other Yucca Mountain database records, you can optionally **Select most recent value no later than**, in which case GoldSim will interrogate the database for the most current version of the source file dated on or prior to that date. If this field is left blank, GoldSim will use the default effective date, as specified in the GS_Parameter table. Details in the format of the Yucca Mountain database are provided in Appendix E.

Locking onto a File

GoldSim allows you to "lock onto" the file that is being referenced by a file element. When you lock onto a file (by checking the **Lock onto this file** option in the File element dialog), the following additional information regarding the referenced file is saved with the element:

- File and path name;
- Date the file was created;
- Date the file was last modified;
- File size; and
- CRC signature.

Once you have locked onto a file, this information is displayed in a tool-tip when you hold your cursor over the filename in the dialog.

The CRC signature is an alphanumeric code that can be used to uniquely identify whether the file contents have changed. When you run a model that has locked onto a file, GoldSim compares the CRC signature of the file with the original signature that was stored. If these are not identical (indicating that the file has been changed), GoldSim displays an error message and will not run the model.

You can unlock a file by clearing the **Lock onto this file** checkbox. Note that if versioning is enabled, whenever a file is locked onto or unlocked, this information is logged with the version.

Read more: Tracking Model Changes (page 963).

Localizing Containers

In most models, element names are forced to be unique. In some cases, however, you may want to have more than one element with the same name. This requirement usually arises when you have a number of similar subsystems in your model.

For example, suppose that you have developed a subsystem (within a Container) that calculates the balance in a bank account as a function of the previous balance, interest rate, deposits and withdrawals. Each of these items would be represented by its own element in the Container. After creating these elements, you decide you would like to track ten other accounts.

In such a case, it would be convenient to create ten copies of the Container (one for each of the other ten accounts). If GoldSim allowed you to just make copies of the bank account Container, however, there would be confusion when you referred to an element by name (e.g., in an expression), since GoldSim would have no way of knowing which copy of the element you were referring to.

GoldSim's solution to this problem is to allow you to localize portions of your model, such that any element names in the local region are hidden from elements outside of that region. A local region is referred to as a ***scope*** in GoldSim. Elements with the same name cannot exist within the same scope. Elements can, however, have the same name if they are located in different

scopes. You change the scope of an element by localizing the Container in which it is located. Unless you specifically "expose" an element, the contents of a local Container can only be "seen" and hence referenced by other elements inside the Container.

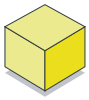
The example model LocalizedContainers.gsm in the General Examples/Containers folder of your GoldSim directory contains an example of the use of Localized Containers.

Localizing a Container

All Containers in a model are either global or local. By default, new Containers that you insert into a model are global. You can localize a Container in two ways:

- By selecting **Localize** from the context menu for the element in the graphics pane or a browser (accessed via a right-click); or
- By selecting the **Localization** checkbox on the properties dialog for a Container.

You can recognize a localized container in four ways:



Localized Container

- The "+" in the upper left hand corner of the Container's symbol in the graphics pane is red (instead of black).
- The icon for the Container in the browsers is a closed box (rather than an open box).
- The default symbol for the Container in the graphics pane is a closed box (rather than an open box).
- The tool-tip for a localized container will display "Localized".

In addition, the properties dialog for the Container will indicate that it is localized.

If a Container is localized, then *internally* it is like a new model: all of the elements within it must have unique names. The elements within the localized Container, however, can share a name with elements outside of the Container.

Note: Like global Containers, localized Containers can be nested, so that a localized Container (referred to here as the parent) can itself contain a localized Container (referred to here as the child). In such a case, the scope of the parent localized Container does not extend into the child localized Container. Hence, an element ID within the child could be the same as one in some other location within the parent.

Read more: Nesting Localized Containers (page 890).

In addition to manually localizing a Container, GoldSim will also automatically localize Containers under some circumstances. In particular, whenever you paste a Container from one portion of your model (or one model) to another, GoldSim will automatically localize it if the names of elements inside the pasted Container conflict with those at the location where it is being pasted. (When it does this, it will warn you with a message.)

Referencing the Contents of a Localized Container

By design, the contents of a localized Container can only be "seen" (and hence referenced) by other elements inside the Container. Therefore, if you place an element called A in a localized Container called C, and then try to reference A in an expression (outside of C), GoldSim will report that it cannot find A (since it is "hidden" within the scope of the localized Container).

Eventually, however, you will need to pull information out of the localized Container to another portion of your model. GoldSim allows you to propagate outputs outside of localized Containers by *exposing* them as if they were outputs of the localized Container itself. Once an output is exposed, it is referenced in expressions by using the form ContainerID.OutputID.

For example, suppose you had a localized Container L, containing an Expression element named X. Assume further that another element named X existed outside of Container L. If you exposed X on the localized Container L, then you could access it from other Containers (outside of L) by referencing it as L.X in expressions.

There would be no confusion with the other element named X in the model, since you would be referring to the X within Container L as L.X (not simply X). Within Container L, however, you could access X in the normal way, by entering X directly in expressions.

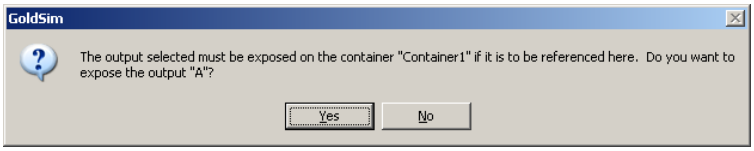
Exposing an Output on a Localized Container

An output can be exposed on a localized Container in four ways:

1. Whenever you localize a Container, any outputs which are already referenced outside of the Container are automatically exposed.

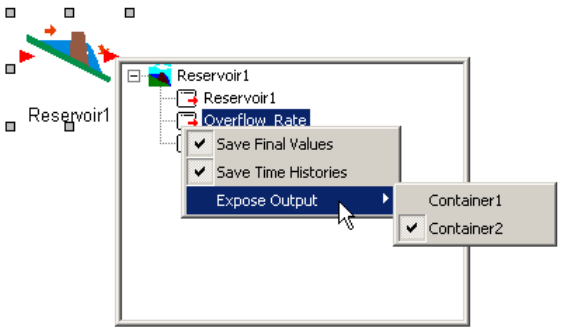
2. If you use the Link Cursor to create a link from an output within a localized Container to a location outside of the Container, the output is automatically exposed.

3. If you use the Insert Link dialog (accessed by selecting **Insert Link…** from the context menu of an input field) to create a link from an output within a localized Container to a location outside of the Container, you will be prompted with a message asking if you want to expose the output:



   If you select **Yes**, the output is exposed.

4. You can manually expose an output on a Container using the context menu for the output (accessed by right-clicking on the output in a browser or an output interface).



   The dialog lists all localized Containers containing the element (i.e., if the element is contained within nested localized Containers, all of the localized Containers are shown). The output can be exposed on any (or all) of the localized Containers in which it resides by clicking on the

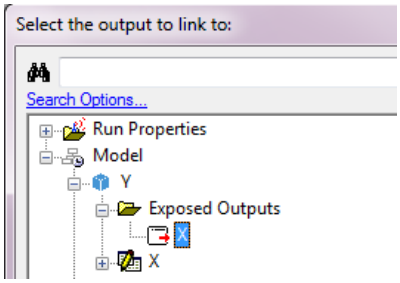
Container name. If the output is already exposed on a particular Container, a check will appear next to the Container name.

Once an output is exposed, it remains exposed even if the element to which it was originally linked is deleted. The only way to unexpose an output is to access the context menu for the output, and under the Expose Output option, clear the check mark (by clicking on it) for the Container.

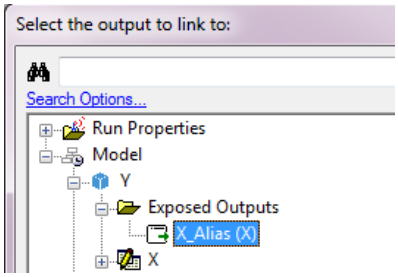
Note: Because moving an element is implemented as a cut and paste operation, and this operation effectively deletes and then recreates the links, if the move involves localized Containers, situations can arise where GoldSim will not be able to recreate all of the links. This can occur, for example, if you move an element into a localized Container such that elements referencing its outputs can no longer see them, or if you move an element outside a localized Container that references an output inside that Container. In such cases, you will have to expose the outputs (and press F9) in order to recreate the links.

Read more: Moving Elements Between Containers (page 106).

Once an output is exposed on a Container, it is displayed within an *Exposed Outputs* folder within the Insert Link dialog (accessed by right-clicking in an input field and selecting **Insert Link…**. For example, suppose that the variable X was exposed on the Container Y. If you selected **Insert Link…** from an input field outside of the Container, you could select X from the Exposed Outputs folder:



Note that if the exposed output has a defined alias, this is shown first, followed by the element name in parentheses:



Read more: Defining an Alias for an Exposed Output (page 890).

Similarly, when typing output names into an input field, GoldSim's link suggestion feature makes allows you to easily enter exposed outputs. In particular, if you wish to link to an exposed output of a localized Container, you can do so by typing a period after the name of the Container in the input field. The suggestion box will then list any exposed outputs that exist for that Container.

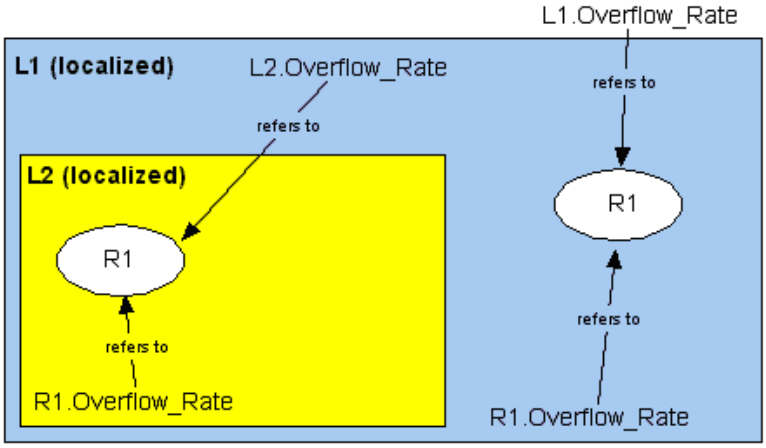
Read more: Displaying Link Suggestions in Input Fields (page 89).

Nesting Localized Containers

Localized Containers can be nested, so that a localized Container (referred to here as the parent) can itself contain a localized Container (referred to here as the child). In such a case, the scope of the parent localized Container does not extend into the child localized Container.

To illustrate this, consider the following example. Suppose you had a localized Container L1, containing a second localized Container L2. A Reservoir element named R1, with an output (among others) called Overflow_Rate, exists in L2. Because this Reservoir is within a localized Container, a second Reservoir element with the same name could exist in L1.

If the Overflow_Rate for the Reservoir in L2 was exposed on L2, and the Overflow_Rate for the Reservoir in L1 was exposed on L1, these outputs would be referenced as shown schematically below:



As indicated in the diagram,

- The Overflow_Rate for the Reservoir within L2 would be referenced as:
  - R1.Overflow_Rate from within L2; and
  - L2.Overflow_Rate from within L1 (but outside of L2).
- The Overflow_Rate for the Reservoir within L1 would be referenced as:
  - R1.Overflow_Rate from within L1 (but outside of L2); and
  - L1.Overflow_Rate outside of L1.

Hence, if you were to reference R1.Overflow rate from an expression within L1, it would link to a different element than if you were to do the same from an expression within L2.

Note: In the example shown above, it is not possible to reference the Overflow_Rate for the Reservoir within L1 from within L2. In order to do so, you would need to first rename one of the two Reservoir elements.

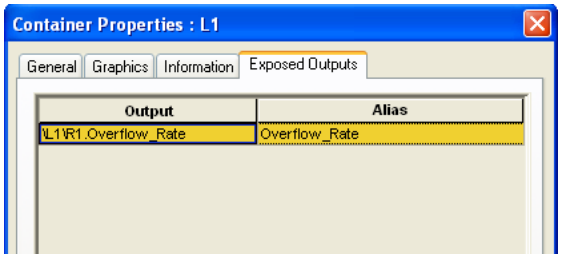
Read more: Search Logic for Linking to an Output Present in Multiple Scopes (page 892).

Defining an Alias for an Exposed Output

There are some situations where referencing an exposed output of a localized container might be ambiguous. In particular, suppose you exposed the Overflow_Rate from the Reservoir in L2 on Container L1. How would you then

reference this output from outside of L1? L1.Overflow_Rate is already used (it refers to the output from the Reservoir in L1). The same problem would occur if you added another Reservoir element with a different name (e.g., R2) to L1 and exposed the Overflow_Rate on L1. Again, L1.Overflow_Rate is already used and is not available.

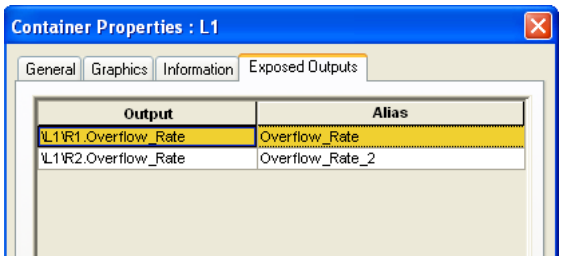
GoldSim solves this potential conflict by defining an alias for each exposed output. Whenever at least one output is exposed on a Container, an **Exposed Outputs** tab is added to the Container's property dialog:



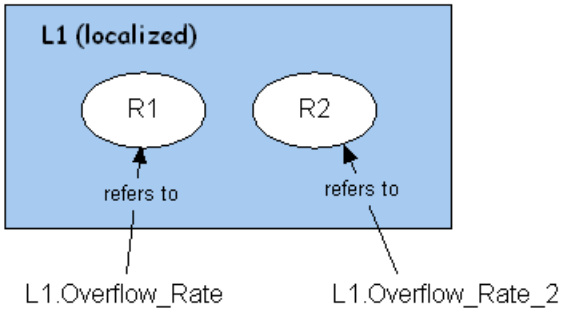
If you hold the cursor over the Output field, a tool-tip showing the full path to the output is displayed.

The first column on this tab lists the full path of the output (which uniquely identifies it). The second column contains an ***alias***. By default, the alias is the output name. When you reference an exposed output outside of a localized Container, it is not actually referenced as ContainerID.OutputID. Rather, it is referenced as ContainerID.Alias.

If you expose an additional output which has the same output name, GoldSim automatically appends a number to the alias.



In the example above, if you exposed the Overflow_Rate from Reservoir R2 on Container L1 *after* exposing the Overflow_Rate from Reservoir R1 on L1, then outside of L1 R2.Overflow_Rate would be referenced as L1.Overflow_Rate_2:



Note: The aliases automatically created by GoldSim for outputs with the same name depend on the order in which the outputs are exposed. The alias of the first output with a particular name that is exposed is always the output name. GoldSim then adds numbers (beginning with 2) to each additional output with the same name that is exposed on the Container.

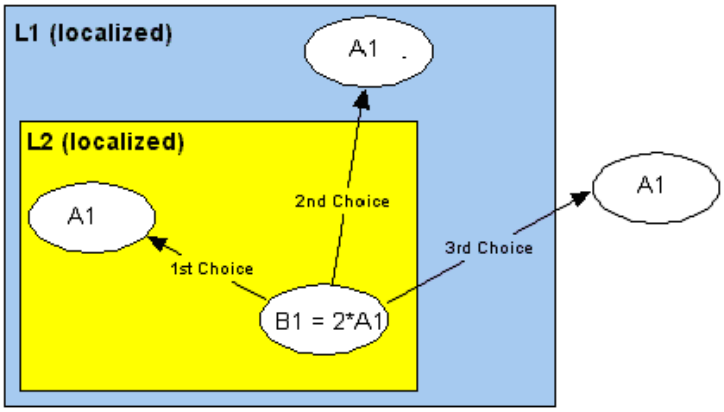
Although GoldSim automatically creates aliases, you can manually edit an alias if you wish (by clicking on it in the dialog). The same rules that apply for element names apply for aliases: Aliases can only include letters, numbers, and the underscore ("_") character. They cannot include spaces and must begin with a letter. Furthermore, for a given Container, no two exposed output aliases can be the same.

Search Logic for Linking to an Output Present in Multiple Scopes

Although the outputs of an element in a localized Container can be referenced outside of the Container only if the output is exposed on that Container, the inputs to the element can come from any location in the model. That is, a localized Container is analogous to a two-way mirrored glass: you cannot look into the Container, but you can look out of it.

In cases where multiple outputs with the same name exist in a model, however, it is important to clearly identify the search logic used by GoldSim to link to outputs.

Consider the following example. Localized container L2 is contained within localized Container L1. Both L1 and L2 contain an element (with a primary output) named A1. A third element named A1 also exists outside of L1. Within L2, you create an Expression element (named B1), and specify its definition as 2*A1. Which of the three A1 elements does GoldSim create a link to?



The schematic above illustrates the search logic GoldSim would use to create the link. It would first look within its own scope (the local scope of B1). If it finds A1 there, it creates the link. If it does not find A1 in its local scope, it moves one level up in the scope hierarchy (in this case within L1) and looks for it there. If it finds A1 there, it creates the link. If it does not find A1 at that level, it continues to search upward through the various scopes that it has access to until it finds an A1, or exhausts the search (and fails to complete the link).

Warning: In the example shown above, if A1 exists in L2, it is impossible to access the outputs of the other A1 elements in the model from within L2. That is, B1 will always try to link to the "nearest" A1, and there is no way for B1 to reference one of the other A1 elements in the model. Even if you use the Link Cursor or the Insert Link dialog to explicitly link to one of these other elements, GoldSim will ignore this and link to the "nearest" A1. It will try to do this even if the output of the nearest A1 is incompatible with input requirements (and one of the other A1 has a compatible output). If you needed to access one of the other elements, you would need to change its name (to something that did not exist inside L2).

Globalizing a Container

You can convert a localized Container to a normal (global) Container in two ways:

- By selecting **Globalize** from the context menu for the element in the graphics pane or a browser (accessed via a right-click); or
- By clearing the **Localization** checkbox on the Container's property dialog.

When you do this, GoldSim first checks to ensure that none of the elements in the Container will conflict with (i.e., have the same name as) other elements in the scope into which the Container's contents will be placed when it was globalized. If it finds a conflict, it will not globalize the Container, and will issue a warning message. You will not be able to globalize the Container until all of the conflicts are eliminated (i.e., until all conflicting elements are renamed).

Cloning Elements

In some situations, you may wish to have elements in your model that always have identical definitions. You could accomplish this by copying an element to various locations in your model, but if you ever needed to change the definition of the element, you would need to make the change to all the copies, and this could be time-consuming and error-prone.

GoldSim addresses this issue by allowing you to create multiple ***clones*** of an element. When you create clones of an element, all of the clones behave identically: if you change one of the inputs to a clone, the same input is automatically changed for all of the other clones.

Clones are often used in conjunction with localized Containers. For example, suppose that you wished to model the salmon populations in each of ten streams. To do this, you might create ten parallel (adjacent) localized Containers, each of which represented a different stream. Although the inputs differ, let's assume that the basic algorithm used to compute the salmon population as a function of time was the same for each of the ten streams.

One way to build this model would be to create the elements representing the algorithm and copy them to each of the ten Containers. If, however, you were expecting to frequently change and modify the algorithm (e.g., as more data became available), it would become cumbersome (and error-prone) to edit the elements in all ten Containers whenever you wanted to make a change.

Clones provide an easier way to build and maintain such a model: You would simply create ten sets of *clones* (rather than *copies*) of the elements defining the algorithm and place them in each of the ten localized Containers. Then if you wanted to modify the algorithm, you would need only to change the elements in

one of the Containers and the corresponding clones of those elements would automatically be changed in all of the others.

Note that use of clones does not imply that all clones produce the same output result; it only implies that they have the same definition. For example, a cloned Expression element might have the definition 2*A. All clones of the element would indeed have the same definition, but if the clones were each located in a different localized Container, then A in one Container could have a different value than A in another Container (i.e., each localized Container could have its own element A). As a result, the output of the Expression would be different in each Container.

Creating Clones

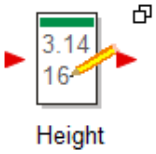
You create clones of an element by right-clicking on the element you wish to clone in the graphics pane or a browser to access its context menu, and selecting **Clone Element**.

When you do so, you will then be presented with a dialog containing all of the Containers in your model. Using this dialog, you must then select the Container into which you wish to place the clone.

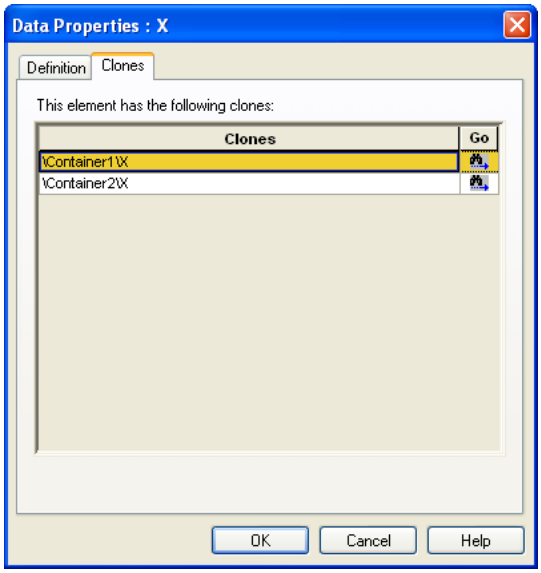
If you select a Container in the same scope as the element you are cloning (or an existing clone of the element), GoldSim will ensure that the names of none of the clones conflict by adding a number to the end of the new clone. If you select a Container in a different scope, the cloned element will have the same name as the element being cloned.

All clones are "equal". That is, the original element is no different than the clone which was created from it. Both are clones, and if you change an input to one, the same input in the other is automatically changed accordingly.

You can always recognize a clone in the graphics pane, because a small symbol (two overlapping boxes) will be present in the upper right-hand corner of the image:



In addition, within the the properties dialog of a cloned element, you will notice an additional tab, labeled **Clones**.



The **Clones** tab lists all of the clones of the element. There is no limit on the number of clones of an element that can be created.

Pressing the **Go** button in this dialog jumps directly to the selected clone.

Moving and Copying Clones

When you move a cloned element (by right-clicking and selecting Move To…), the element remains a clone after it is moved.

However, if you copy and paste a cloned element, the pasted element is not a clone.

Element Properties that are Not Cloned

If you change an input for one clone, it is automatically changed for the other clones. This is the case for standard inputs, as well as other properties, such as checkboxes.

There are, however, several exceptions to this rule:

- Names (element IDs) are never cloned. Each clone can have a unique name.
- Descriptions are never cloned. Each clone can have a unique description.
- Save flags (i.e., Save Final Values and Save Time Histories) are never cloned. You must separately select whether you wish to save results for each clone.
- The graphical properties of an element (e.g., size, color, and symbol used in graphics pane) are not cloned. Each clone can have a different appearance.
- The random seed of Stochastic elements is not cloned. Therefore, cloned Stochastic elements have different seeds (and hence will have different sampled values).
- A number of elements (e.g., Reservoirs) provide an input field that only accepts Discrete Changes. These fileds are followed by a button that allows you to more easily add multiple Discrete Changes. If an element is cloned, this button is disabled (such that the only way to add multiple Discrete Changes is by separating them in the input field using a semi-colon).

Elements that Cannot be Cloned

Six element types cannot be cloned:

- Result elements (including Decision Analysis elements);
- Script elements;
- SubModels;
- CellNet Generators;
- File Elements; and
- Dashboards.

In addition, if a Data element is defined to provide Scenario Data, it cannot be cloned.

Read more: Scenario Data: Defining Inputs for Different Scenarios (page 465).

No option is provided to clone these elements. Moreover, if they are present in a Container that you want to clone, you will not be able to clone the Container. You also cannot insert any of these elements into a previously cloned Container.

Read more: Cloning Containers (page 896).

Freeing a Clone (Decloning)

In some situations, you may want to "declone" or "free a clone" (i.e., change it to a "normal" element). You can do this by right-clicking on the clone you wish to declone in the graphics pane or a browser to access its context menu, and selecting **Free Clone**.

In either case, the element will be changed into a "normal" element: it will no longer be a clone (and the Clone tab will no longer appear in the properties dialog for the element).

Note that if the element was one of only two clones before it was freed, the other remaining clone would also be automatically freed (since it would have no other clones).

Similarly, if you delete a clone which was one of only two clones before it was deleted, the remaining clone is automatically freed.

Cloning Containers

In addition to cloning individual elements, you can also clone an entire Container. Before cloning a Container, it must first be localized. Once a Container is cloned, it cannot be globalized.

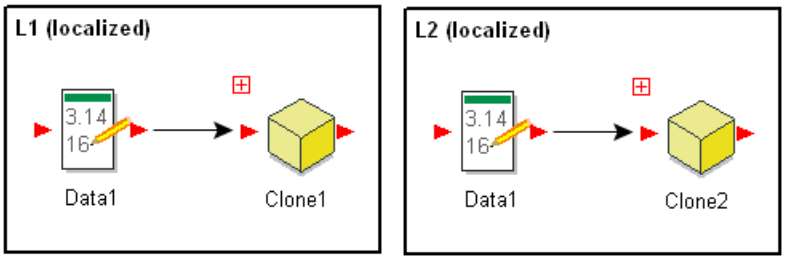
When you clone a Container, all of its contents are also automatically cloned. Any change you make to any element within one cloned Container is made to all of the clones. In particular,

- When you create, paste or delete an element in one cloned Container, it is automatically created, pasted or deleted in all of the cloned Containers.
- When you move an element between Containers inside of a cloned Container, it is automatically moved within all of the cloned Containers.
- When you move an element into a cloned Container, the element is replicated and moved into all of the cloned Containers.
- When you move an element out of a cloned Container, that copy is moved and all of the clones of the element in the other cloned Containers are deleted.

Several other points regarding cloned Containers should also be noted:

- The names of all elements within a cloned Container are also cloned. (Recall that cloned elements which are not inside a cloned Container can have different names).
- You cannot declone an element which is inside a cloned Container.
- When you declone a Container, the elements inside the formerly cloned Containers remain clones. Once the Container is decloned, however, you can individually declone elements within the Container.
- Some elements cannot be cloned (e.g., Dashboards, Scripts, SubModels, Files, Results). As a result, GoldSim prevents your from adding these to a cloned Container, and prevents a Container that contains such elements from being cloned.
- If graphic images are present inside a Container when it is cloned, these are copied to the cloned Containers. If they are subsequently edited, however, the changes are not propagated to the other cloned Containers.
- When you change the positions of elements within a cloned Container (e.g., by dragging an element from one part of the screen to another), the new position is not propagated to the other cloned Containers.

The key to using cloned Containers is to provide unique inputs to them from elements that are *outside of them* (otherwise, all of the cloned Containers would behave identically and produce exactly the same results). This can be done by placing the cloned Containers within separate localized Containers:



In the example shown above, the two cloned Containers (Clone1 and Clone2) are within two separate localized Containers (L1 and L2). Within the cloned Containers, all the equations and elements are, by definition, identical. In this example, it is assumed that within each cloned Container, an element references Data1. Since the clones are in different localized Containers, however, they each link to a *different* Data1 element.

The example model CloningContainers.gsm in the General Examples/Containers folder of your GoldSim directory contains an example of the use of cloned Containers.

Copying Containers with Clones

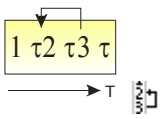
In general, you cannot copy a cloned element to create another cloned element. If you simply copy a clone and paste it elsewhere, the pasted element will not be a clone. The only way to clone an element is to right-click on it and select **Clone Element.**

Under one circumstance, however, you can create a clone by copying and pasting an element. In particular, you can create clones if you copy and paste a Container that contains clones.

When you copy and paste a Container that contains clones, GoldSim uses the following rules when the Container is pasted:

- If the Container contains at least two elements that are clones of each other (cloned sisters), the pasted Container will also treat those elements as clones. Note, however, that they will not be clones of the elements in the original Container. For example, if Container1 contains elements A and B that are sister clones, and Container1 is copied and pasted (and the pasted Container is named Container2), A and B inside Container2 will be clones of each other, but they will not be clones of the original A and B in Container 1. The rule here is simple: when you copy and paste a Container with clones, under no circumstances can you increase the size of a set of existing clones. You can only make a copy of a set (forming a new set of clones).
- If the Container contains only one clone (with the other clones being outside of the copied Container), the element will not be a clone in the pasted Container.

Referencing an Output's Previous Value



There are some situations in which you may want to reference the *previous* (as opposed to the *current*) value of an element's output. For example, you may want to reference the amount of money in an account (represented by a Reservoir) one day ago, as opposed to at the current time. GoldSim allows you to reference an output's previous value by using a Previous Value element.

A previous value element outputs the value of its input from the *previous update* of the model. Usually, a model is only updated every timestep, so that the previous update refers to the previous "scheduled" timestep specified in the Time Phase settings of the Simulation Settings dialog.

Read more: Setting the Basic Time Options (page 413).

In some cases, however, GoldSim actually updates the model between specified (scheduled) timesteps. In such a case, the previous value is not necessarily the previous scheduled timestep, and may return a value between the current scheduled timestep and the previous scheduled timestep.

Read more: Understanding Timestepping in GoldSim (page 415).

There are basically three instances in which a Previous Value element can be useful:

- Occasionally, you may actually want to directly reference the previous value of an output (as in the example of an account mentioned above). Generally, however, you should be careful when doing so, and in most cases it is poor modeling practice, as it ties your model to the timestep length. For example, if you were interested in obtaining the value for an output from 3 days previous, and to do so you used a Previous Value element, you would need to take care to always use a 3 day timestep. In such a case, rather than choosing a 3 day timestep and referencing its previous value using Previous Value element, it is almost always better to use an Information Delay with a Delay Time of 3 days.

  Read more: Information Delay Elements (page 292).

- In some situations, you may wish to simulate a static or dynamic process in which the variables in the loop are coupled such that they respond instantaneously to each other, and there are no time lags. In GoldSim, these are referred to as ***recursive loops***, and they are conceptually different from feedback loops. The output of a Previous

Value element can be used to provide an approximation to the current value of an output so that coupled equations can be solved explicitly.

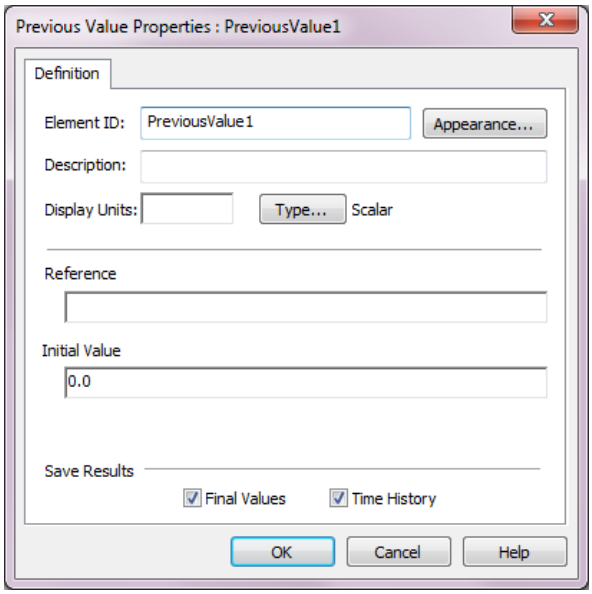
Read more: Creating Recursive Loops Using Previous Value Elements (page 901).

- GoldSim provides the ability to carry out looping (iterative) calculations at each update (at each point in time) within a Container. In these special Containers, a calculation is repeated until a specified number of loops is completed or a particular condition is met. In most cases, the condition will likely involve comparing the output of the looping calculation to the previous loop's output (e.g., determining if ($X_{now} - X_{prev}$) < some tolerance value). Previous Value elements can be used to facilitate such a calculation.

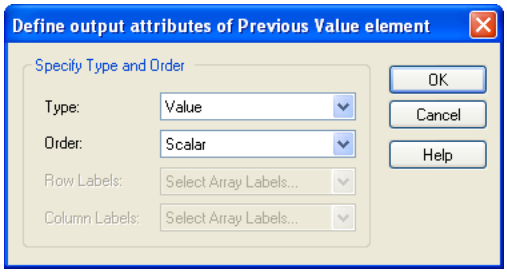
  Read more: Using Looping Containers (page 904).

Inputs and Outputs to a Previous Value Element

The property dialog for a Previous Value element looks like this:



When defining a Previous Value element, your first step should be to specify the type of output for which you want to compute a previous value. You do this by pressing the **Type…** button, which will display the following dialog:



By default, the output of a new Previous Value element is a scalar, dimensionless value. However, a Previous Value element can accept as an input (and hence output) one of four types of outputs (as specified in the **Type** field in this dialog): Values, Conditions, Discrete Change Signals and Discrete Event Signals. If you select Value, Condition, or Discrete Change Signal, the output can also be specified as a vector or a matrix.

The **Display Units** determine the dimensions of the input to and output from the Previous Value element, and are only applicable if the output is a Value or a Discrete Change Signal (otherwise the field is grayed out).

The **Reference** is the value for which you want to compute a previous value.

Note: If the **Reference** is a Discrete Event Signal or a Discrete Change Signal, it can accept multiple discrete signals. This is indicated in the input field by separating the individual discrete signals by semi-colons (e.g., Change1; Change2; Change3).

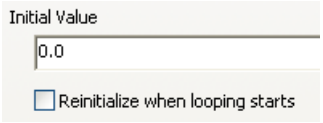
If the **Reference** is a Value or a Condition, in order to compute the output of a Previous Value element, GoldSim must know what value to use at the beginning of the simulation. This is specified within the **Initial Value** field.

Note: The **Initial Value** must be a number or a link from a static variable (e.g., a constant Data element, a Stochastic, or an Expression that is a function of constant elements).

Obviously, the **Reference** and the **Initial Value** inputs to the Previous Value element must have the same attributes (type, order and dimensions) as specified in the attributes dialogs (accessed via the **Type…** button).

Note: If you specify the output type as a Discrete Event Signal, the Reference field will also accept Discrete Change Signals. However, the information in the signal (Instruction and Value) will be lost when it is output by the Previous Value element.

If you place a Previous Value element inside a looping Container (and the Reference is a Value of a Condition), the element has an additional property (**Reinitialize when looping starts**) that controls how the Initial Value is treated between loops:



Read more: Using Looping Containers (page 904).

A Previous Value element has two outputs:

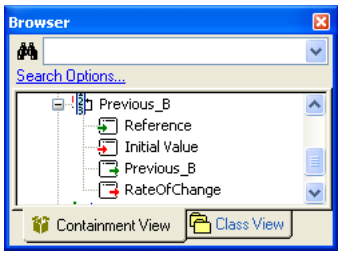
- The previous value of the Reference; and
- The rate of change of the Reference (i.e., the derivative).

The previous value is the primary output.

Note: The rate of change output is only available if the Reference is a Value.

Viewing a Previous Value Element in the Browser

The browser view of a Previous Value element is shown below:



As can be seen, the browser view shows the two inputs (Reference and Initial Value) and the two outputs (the previous value itself, which has the same name as the element, and the RateOfChange).

Note: Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

Creating Recursive Loops Using Previous Value Elements

In some situations, you may wish to simulate a static or dynamic process in which the variables in the loop are coupled such that they respond instantaneously to each other, and there are no time lags. In GoldSim, these kinds of circular systems are referred to as recursive loops (and are conceptually different from feedback loops).

Read more: Evaluating Feedback Loops (page 314).

A simultaneous system of equations is an example of such a system. Consider the following example:

$$B = 4 - A$$

$$A = (6 - B)/3$$

This is a circular system, because A is a function of B and B is a function of A. To compute A and B, it is necessary to solve this coupled system of equations. In this particular case, through simple substitution, you can convince yourself that the solution to these equations is A =1 and B=3.

If you encounter a system such as this in one of your models, you can handle it in one of two ways. First, you could solve the system of equations directly either prior to running the model (e.g., using substitution), or dynamically while running the model (e.g., using an External element or GoldSim's matrix functions to solve the appropriate equations).

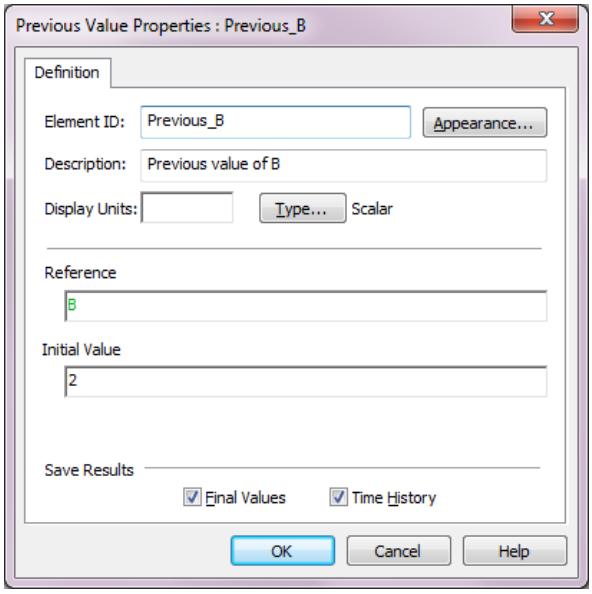
For simple systems such as the example above, this is obviously the correct approach. Note, however, that for many complex models (e.g., non-linear coupled equations), the solution could be very computationally intensive, requiring an iterative approach.

GoldSim therefore will allow you to specify such a circular system and solve the equations in an approximate manner. It does this by assuming, at every timestep, that the current value of one of the variables can be approximated by the value of that variable at the previous timestep. To solve the simple system outlined above, we would define the two equations in GoldSim as follows:

$$B = 4 - A$$

$$A = (6 - \text{Previous\_B})/3$$

In the second equation, Previous_B is the output of a Previous Value element whose input is B:



By doing so, the equations can be solved explicitly (A is solved first as a function of the previous value of B, and then B is solved as a function of A). In effect, GoldSim uses the timestepping algorithm to iterate to a solution, as illustrated in the table below:

Timestep	B at previous timestep	A	B
0	2	1.333	2.667
1	2.667	1.111	2.889
2	2.889	1.037	2.963
3	2.963	1.012	2.988
4	2.988	1.004	2.996
5	2.996	1.001	2.999
6	2.999	1.001	3.000
7	3.000	1.000	3.000

In this case, GoldSim iterated to a very good solution within 2 or 3 timesteps, and an almost exact solution in 7 timesteps. In order for this to work, however, it was necessary to specify the value of "B at the previous timestep" when time = 0. That is, it was necessary to define an Initial Value for B. For this example, an Initial Value of 2 was assumed.

This simple example model illustrating the use of Previous Value elements (PreviousValue.gsm) can be found in the General Examples folder in your GoldSim directory.

Note: A more effective way to solve this particular problem would be to embed the system of equations in a looping Container. In such a case, the iteration would not require any timestepping (the iteration would occur within a single timestep). In fact, looping Containers are one of the most important uses for Previous Value elements. This is illustrated in the file LoopingContainers.gsm, which can be found in the General Examples/Containers folder in your GoldSim directory. Alternatively, you could solve such a system of equations using a Script element. This is illustrated in the file Script.gsm, which can be found in the General Examples folder in your GoldSim directory.

Read more: Using Looping Containers (page 904); Script Elements (page 803).

When using Previous Value elements outside of looping Containers to represent recursive systems, you need to take special care to ensure that the length of the timestep you are using is small relative to the timescale over which the system is changing. If the system you are simulating is changing rapidly relative to the timestep, your computed result will be inaccurate. That is, if you are using the previous value of a variable as an approximation to the current value, and the previous value is in fact a bad approximation of the current value, then the computed result will be inaccurate (since it takes several timesteps to converge on the correct solution). Selecting a timestep for these kinds of simulations is discussed in further detail in Appendix F.

It is worth reiterating that recursive loops are conceptually different from feedback loops. Feedback loops represent dynamic processes in which the variables in the loop do not respond instantaneously to each other. It takes time for a signal to propagate through a feedback loop. That is, the response among the variables in a feedback loop necessarily involves time lags. GoldSim allows you to create looping systems like this without having to use Previous Value elements. In order to do so, however, the loop must meet one requirement: it must contain at least one ***state variable***.

Read more: Understanding State Variables in GoldSim (page 309).

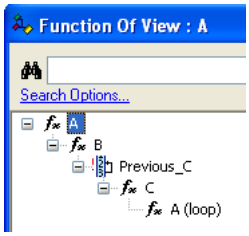
Recursive loops, on the other hand, represent dynamic processes in which the variables in the loop respond instantaneously to each other. That is, they are coupled. Simulating systems like these in GoldSim requires approximating the solution by using Previous Value elements.

Finding Recursive Loops

When using the Function Of or Affects View on an element that is within a feedback or recursive loop, GoldSim will stop building a branch of the dependency tree as soon as an item is repeated (and it will mark this as a "loop").

Read more: Viewing Element Dependencies (page 117).

For example, if A is a function of B, B is a function of the previous value of C, and C is a function of A, the Function Of View for A would look like this:



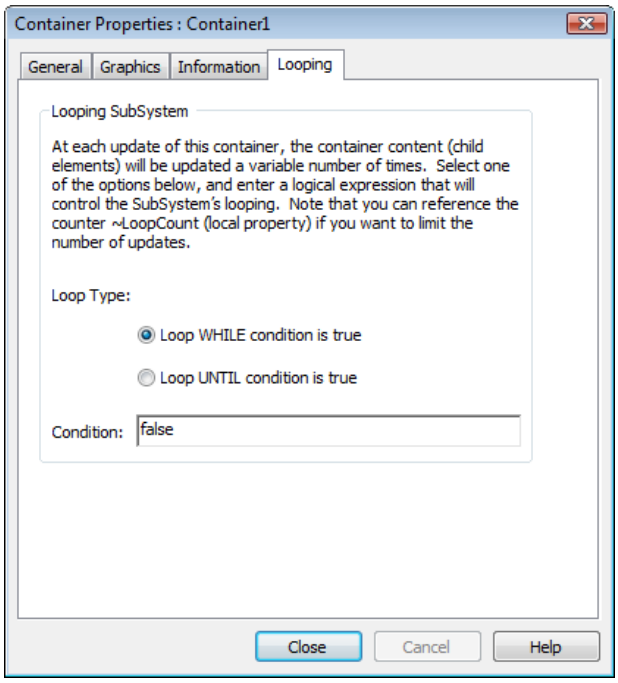
When using the Find function (**Ctrl+F**) in a Function Of or Affects view, one of the options is to search in Labels. If this is selected, and you enter "loop", GoldSim will find the next loop in the list (since it will find the word "loop" in the label for the element). When using this search method, GoldSim will find all the loops (both feedback loops and recursive loops) in the model.

Read more: Finding Elements (page 116).

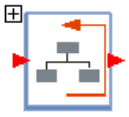
Using Looping Containers

In some models, you may want to carry out an iterative calculation at each timestep. This might be useful, for example, if you have a coupled system of equations that must be solved every timestep by iterating.

You can define a Container as a looping Container by selecting the **Looping Capability** feature in the Container dialog. When you do so, a **Looping** tab is added to the Container dialog:



Looping Containers are represented in the graphics pane as follows:



Note: When you specify a Container as having **Looping Capability**, you cannot also define an **Internal Clock** for the Container (these two options are mutually exclusive).

Note: Looping Containers are useful when the looping calculation necessarily involves multiple elements (e.g., Reservoirs). For a calculation requiring simpler looping requirements (defining an array, or iterating to a solution for a simple equation), a Script element would often provide a more transparent and easier solution.

Read more: Script Elements (page 803).

Warning: When you specify a Container as a looping Container, the **Treat as SubSystem** feature is also automatically selected (and cannot be deselected unless you first turn off **Looping Capability**). That is, a looping Container, by definition, is treated as a SubSystem. Because a looping Container is treated as a SubSystem, this puts certain limitations on how these Containers can be used.

Read more: Treating a Container as a SubSystem (page 139).

Controlling the Number of Loops in a Looping Container

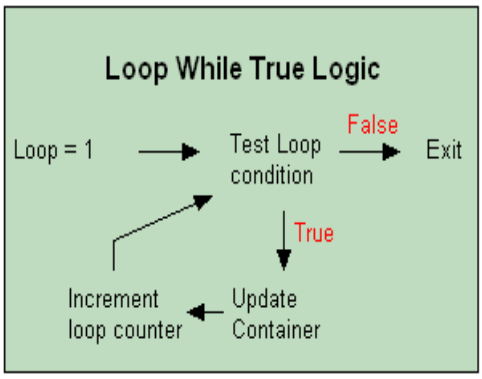
Looping Containers are intended to be used to carry out an iterative calculation at each timestep. The number of loops carried out is controlled by specifying a condition (i.e., loop while or until a specified condition is met).

You can choose between two loop types:

Loop WHILE condition is true. In this case, the loop is carried out as follows:

1. At the beginning of the calculation, the loop counter is set to 1.
2. GoldSim tests the **Condition**.
3. If the Condition is False, the looping calculation is assumed to be complete.
4. If the Condition is True, the contents of the Container are updated and the loop counter is incremented.
5. Go to Step 2.

This is illustrated schematically below:

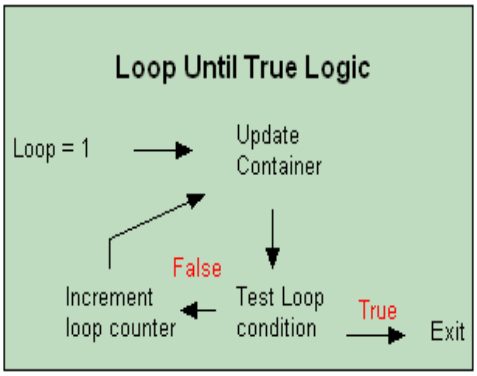


Loop UNTIL condition is true. In this case, the loop is carried out as follows:

1. At the beginning of the calculation, the loop counter is set to 1.
2. The contents of the Container are updated.

3. GoldSim tests the **Condition**.
4. If the Condition is True, the looping calculation is assumed to be complete.
5. If the Condition is False, the loop counter is incremented.
6. Go to Step 2.

This is illustrated schematically below:

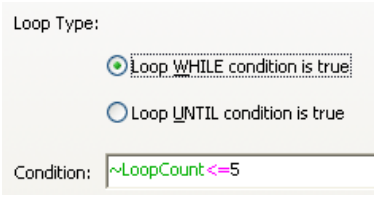


Note: "Loop Until" Containers always update at least once. "Loop While" Containers may not update at all (if the loop condition is not met).

Warning: Because the Condition for a "Loop While" Container must be evaluated before the contents of the Container are evaluated, this Condition can only reference the loop counter or elements inside the Container that have well-defined initial conditions (i.e., state variables)

Read more: Understanding State Variables in GoldSim (page 309)

The loop counter itself can be referenced by any element inside the Container (and within the **Condition** field) as "~LoopCount". For example, if the **Condition** was defined as follows:

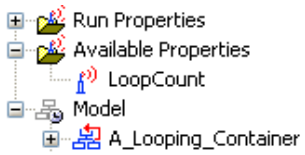


then the Container would complete exactly 5 loops.

As indicated by the way it is referenced, the loop counter is an example of a locally available property. As such, it only has meaning inside the looping Container, and cannot be referenced outside of the Container.

Read more: Understanding Locally Available Properties (page 750).

If you access the Insert Link dialog inside a looping Container (by right-clicking inside an input field of an element within the Container or in the **Condition** field itself), the loop counter appears in a Available Properties folder (displaying all of the locally available properties that can be accessed from that location):



The **Condition** must reference at least one output from within the looping Container (the loop counter is considered to be within the Container). If it does not, GoldSim will display an error message (since otherwise, the **Condition** could not change from loop to loop, and hence has no meaning).

Warning: GoldSim will not exit a loop until the **Condition** has been met, and there is no upper limit on the number of loops that can be carried out. As a result, you should be careful when defining your **Condition** to ensure that you do not define an infinite loop. You can always do so by directly referencing the loop counter (e.g., adding something to the **Condition** such as "OR ~LoopCount > 1000") If it seems that you are in an infinite loop, you should pause the simulation (using the **Pause** button on the Run Controller) and evaluate the loop counter and the condition. You can subsequently stop a simulation using the **Abort** button on the Run Controller.

Understanding How Elements Inside a Looping Container are Updated

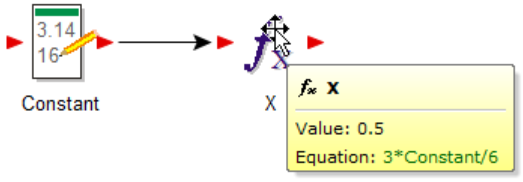
The elements inside the Container are updated as follows during the loops:

1. On the first loop, the elements are moved forward in time to the next scheduled timestep;
2. On subsequent loops, the elements are updated again without advancing time.

Hence, in order for the loops after the first to have any impact on the model, they must contain at least one state variable that changes with each loop.

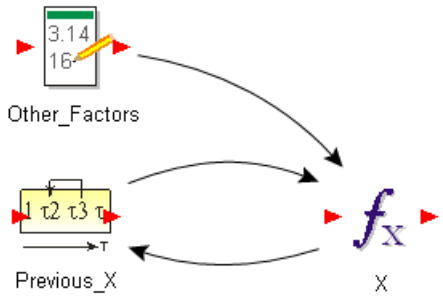
Read more: Understanding State Variables in GoldSim (page 309).

As an example, if a looping Container contained the simple system below, consisting of one Data element and one Expression (neither of which output a state variable), after the first loop, the system would not change at all:



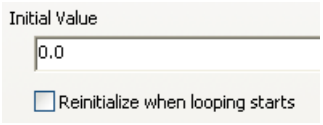
Hence, it would serve no purpose to place a calculation such as this within a looping Container.

The most common application of a looping Container is to use it to carry out an iteration utilizing a Previous Value element. That is, some variable (e.g., X) would be computed each loop based on its value during the previous loop:



The looping system shown here would be updated every loop (changing X , and hence the previous value of X). Typically, you would specify that the looping continue until the difference between X and the previous value of X was within some specified tolerance.

When a Previous Value element is included inside a looping Container, it takes on a special property. In particular, an additional input option is added to the dialog that specifies how the **Initial Value** is to be treated:

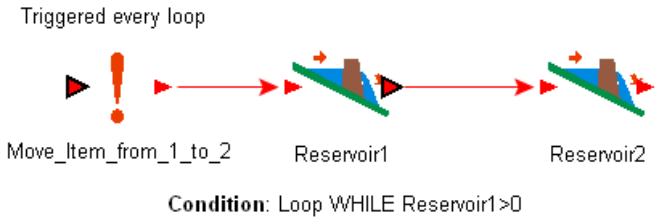


If **Reinitialize when looping starts** is cleared (the default), the Initial Value specified in the field is only used at the beginning of the realization (when the Container is updated for the first time), and never again.

If **Reinitialize when looping starts** is checked, the value specified in the Initial Value field is used for the first loop every time the looping Container begins its calculations (i.e., it does not "remember" the value from the previous timestep).

Read more: Inputs and Outputs to a Previous Value Element (page 899).

Another common application of a looping Container is to use it to carry out multiple discrete transactions in a single timestep. In the simple example below, items are moved from one Reservoir to another (via discrete changes) until the first Reservoir is empty:



Read more: Modeling Discrete Changes to a Reservoir (page 355).

Examples of both of these types of applications (LoopingContainers.gsm) can be found in the Containers subfolder of the General Examples folder in your GoldSim directory.

Viewing and Modifying the Causality Sequence

When you build a model, GoldSim automatically *sequences* the elements in the order that they must be computed. For example if A was a function of B, and B was a function of C, C would be sequenced first, followed by B, followed by A.

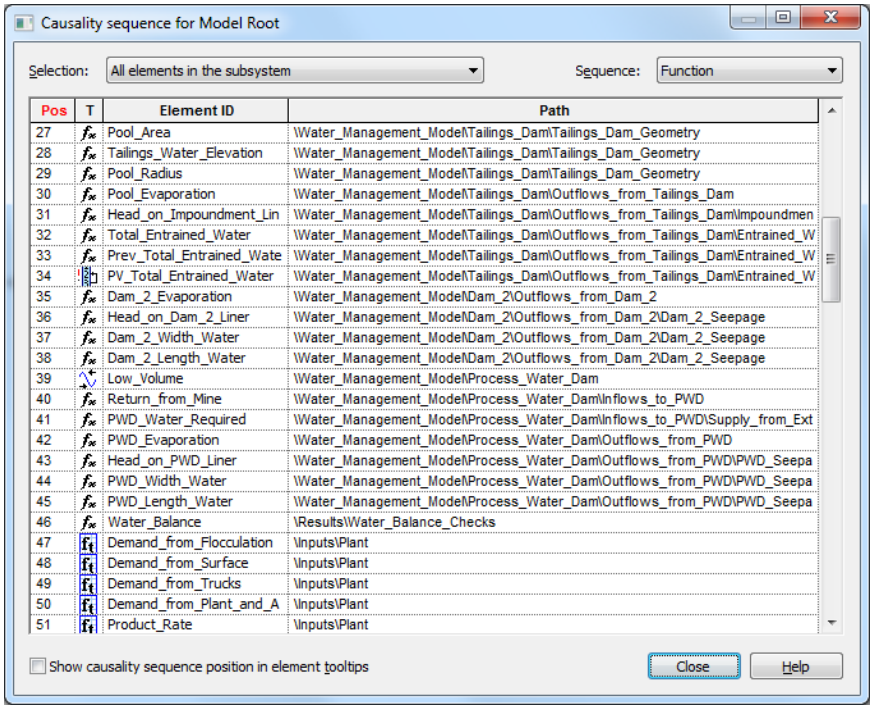
This is referred to as the ***causality sequence***. Although in this simple example, the sequence is obvious, for complex models with looping logic, the causality sequence may not be readily apparent.

Read more: The Causality Sequence and Element Updating (page 311).

In most cases, you need not to be concerned with how GoldSim sequences the elements. However, in some cases (e.g., when simulating systems that include discrete events and looping logic or systems that have competing Resource requirements), expert users may need to understand (and subsequently manipulate) the causality sequence.

Viewing the Causality Sequence

You can view the causality sequence selecting **Model|Causality Sequence** from the main menu (or pressing **F10**). A dialog similar to this will be displayed:



Pos	T	Element ID	Path
27		Pool_Area	\Water_Management_Model\Tailings_Dam\Tailings_Dam_Geometry
28		Tailings_Water_Elevation	\Water_Management_Model\Tailings_Dam\Tailings_Dam_Geometry
29		Pool_Radius	\Water_Management_Model\Tailings_Dam\Tailings_Dam_Geometry
30		Pool_Evaporation	\Water_Management_Model\Tailings_Dam\Outflows_from_Tailings_Dam
31		Head_on_Impoundment_Lin	\Water_Management_Model\Tailings_Dam\Outflows_from_Tailings_Dam\Impoundmen
32		Total_Entrained_Water	\Water_Management_Model\Tailings_Dam\Outflows_from_Tailings_Dam\Entrained_W
33		Prev_Total_Entrained_Wate	\Water_Management_Model\Tailings_Dam\Outflows_from_Tailings_Dam\Entrained_W
34		PV_Total_Entrained_Water	\Water_Management_Model\Tailings_Dam\Outflows_from_Tailings_Dam\Entrained_W
35		Dam_2_Evaporation	\Water_Management_Model\Dam_2\Outflows_from_Dam_2
36		Head_on_Dam_2_Liner	\Water_Management_Model\Dam_2\Outflows_from_Dam_2\Dam_2_Seepage
37		Dam_2_Width_Water	\Water_Management_Model\Dam_2\Outflows_from_Dam_2\Dam_2_Seepage
38		Dam_2_Length_Water	\Water_Management_Model\Dam_2\Outflows_from_Dam_2\Dam_2_Seepage
39		Low_Volume	\Water_Management_Model\Process_Water_Dam
40		Return_from_Mine	\Water_Management_Model\Process_Water_Dam\Inflows_to_PWD
41		PWD_Water_Required	\Water_Management_Model\Process_Water_Dam\Inflows_to_PWD\Supply_from_Ext
42		PWD_Evaporation	\Water_Management_Model\Process_Water_Dam\Outflows_from_PWD
43		Head_on_PWD_Liner	\Water_Management_Model\Process_Water_Dam\Outflows_from_PWD\PWD_Seepa
44		PWD_Width_Water	\Water_Management_Model\Process_Water_Dam\Outflows_from_PWD\PWD_Seepa
45		PWD_Length_Water	\Water_Management_Model\Process_Water_Dam\Outflows_from_PWD\PWD_Seepa
46		Water_Balance	\Results\Water_Balance_Checks
47		Demand_from_Flocculation	\Inputs\Plant
48		Demand_from_Surface	\Inputs\Plant
49		Demand_from_Trucks	\Inputs\Plant
50		Demand_from_Plant_and_A	\Inputs\Plant
51		Product_Rate	\Inputs\Plant

The dialog indicates the order within the sequence (Pos), the element type (T), the Element ID and the full element path (Path).

There are actually two different sequences that can be displayed: the *Static Sequence* and the *Function Sequence*. The Static Sequence consists of those elements that only need to be computed once (at the beginning of the simulation), since they cannot change with time. The Function Sequence consists of those elements that can change as a function of time.

Read more: The Causality Sequence and Element Updating (page 311).

By default, the dialog displays the Function Sequence first (as this is typically of greatest interest). You can switch between these two using the drop-list in the top right corner of the dialog.

You can sort the rows according to any column (ascending or descending) by double-clicking on the column header.

The **Selection** field at the top of the dialog allows you to limit what portion of the sequence is displayed:

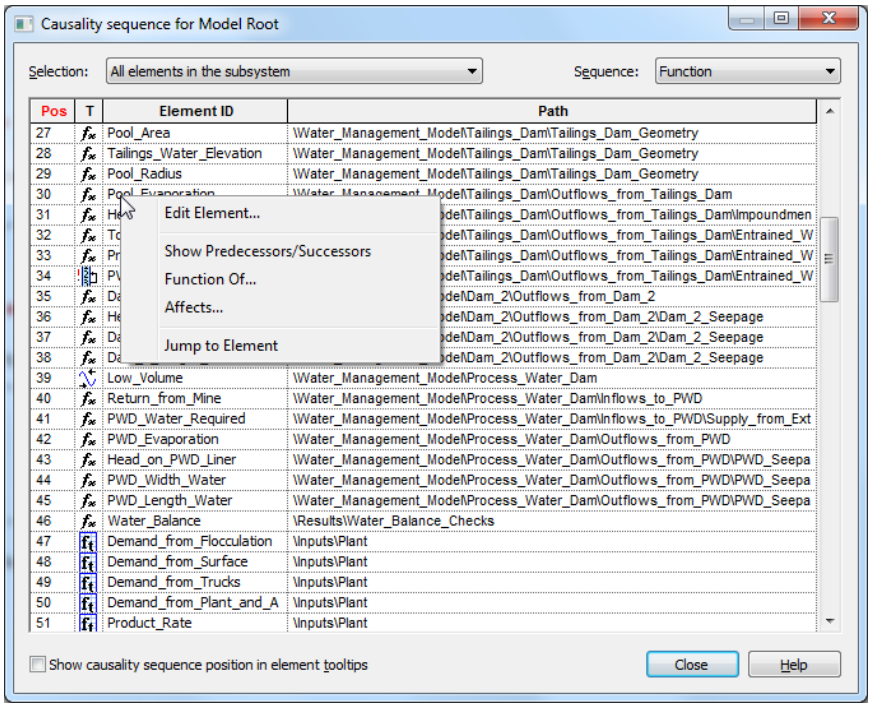
- All elements in the subsystem (the default)

- Elements in current container/child containers
- Elements in the current container only

Note: You can only view the causality sequence for one subsystem at a time. The subsystem that is displayed is the one you were viewing when you displayed the sequence.

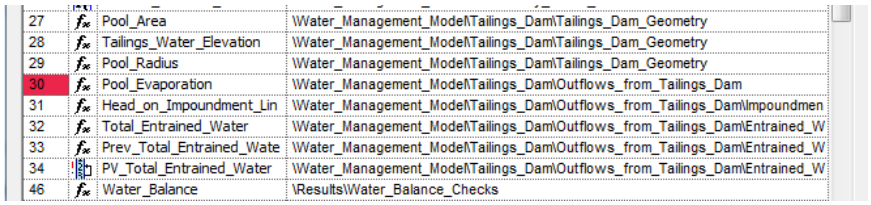
Read more: Treating a Container as a SubSystem (page 139).

Right-clicking on an element in the sequence displays the following menu:



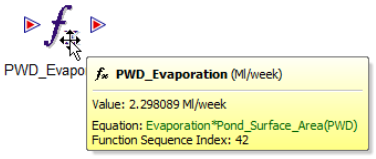
Four of these options are self-explanatory ("Edit Element", "Function Of", "Affects", and "Jump to Element").

"Show Predecessors/Successors" provides a filtered view of the sequence showing only those elements that directly or indirectly either affect the selected element, or are affected by it. Note that when you choose this option, the position of the selected element is marked with a red background (making it easier to see):



"Specify Precedent" allows you to manually affect the causality sequence, and is discussed in detail in the next section.

Checking **Show causality sequence position in element tooltips** adds the position number to tooltips displays in the model:



Note: In some complex looping systems, you may notice that an Element ID in the sequence will be highlighted in red. This indicates that due to the looping nature of the system, GoldSim's first choice for sequencing the element was not possible, and it was necessary to move the element upward in the sequence. GoldSim's preference is to always have referencing elements follow the elements that they reference. When GoldSim is unable to generate a valid sequence using this rule, it selectively moves elements that reference state variables ahead of the elements that calculate the state variables, and flags the moved elements in red. The resulting sequence is not invalid, but it does indicate that the sequence is ambiguous.When you see a system like this, you should take special care to look carefully at the results to determine whether or not the system is performing as you expect it to. If not, you may need to manually change the sequence.

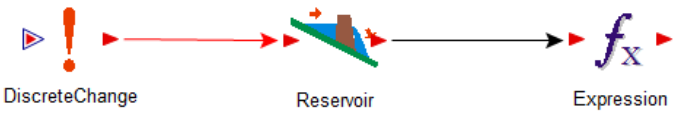
Addressing Ambiguous Causality Sequences

In order to determine how GoldSim is to carry out its calculations, it first must create the ***causality sequence*** for all of the elements. This represents the sequence in which the elements must be logically computed.

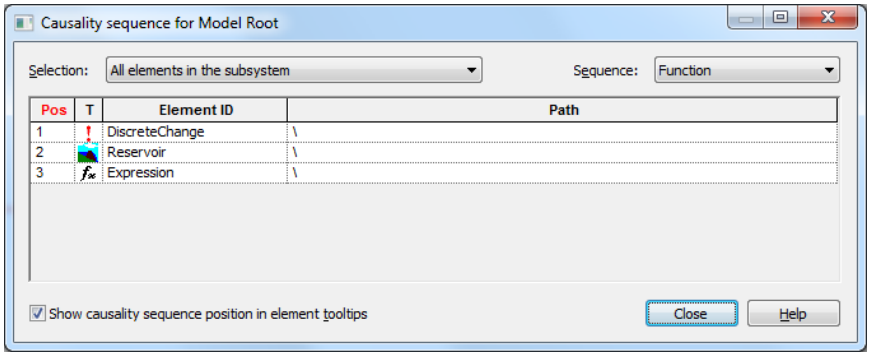
Read more: The Causality Sequence and Element Updating (page 311).

In some models, however, there may be more than one valid way to sequence the system, *and these different sequences may produce different results*. In these cases, GoldSim provides a way to manually force a particular causality sequence to ensure that the model accurately represents your system.

In order to better understand this, let's construct a simple system where the causality sequence is ambiguous:



By default, the causality sequence for this model looks like this:



In order to fully understand this sequence, it is first necessary to recall a special rule regarding a Function Sequence involving Discrete Changes: Discrete Changes are propagated to any affected stock elements instantaneously when the

Discrete Change is updated in the sequence (regardless of where the affected stocks are located in the sequence).

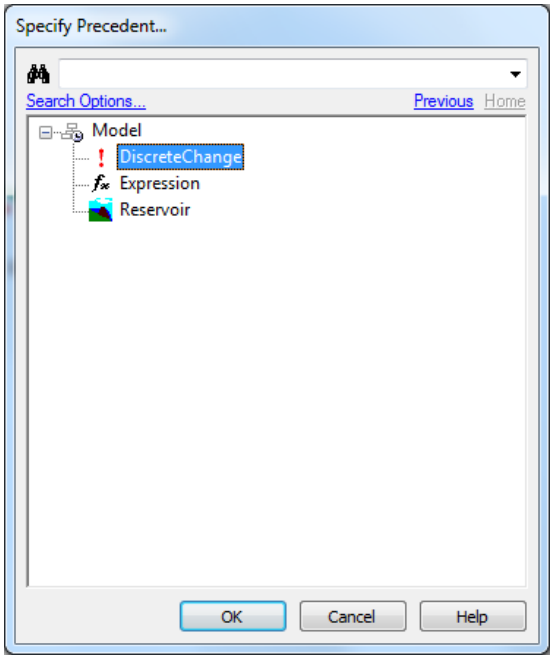
So in the sequence shown above, GoldSim does the following: Every timestep, after changing the system clock, GoldSim first computes the state variables that are intrinsic functions of time. In this example, the only element that falls into this category is the Reservoir, so it is first brought up to the current time. After doing this, GoldSim then computes the Function Sequence. First, it computes the value of the Discrete Change, and instantaneously applies the Discrete Change to the Reservoir (this is actually regardless of where the Reservoir is in the Sequence). Then it computes the Reservoir (i.e., it would store all of its inputs for use in the next timestep, but this would not change its current value). Finally, the Expression is computed.

The reason this system is ambiguous is related to the fact that the Reservoir is actually computed twice. First, it is brought up to the current time (i.e., it solves a time integral). Then, it is updated a second time within the Function Sequence (it is modified by the Discrete Change). The ambiguity here arises from the fact that it is not completely clear when you might want to compute the Expression. There are two options:

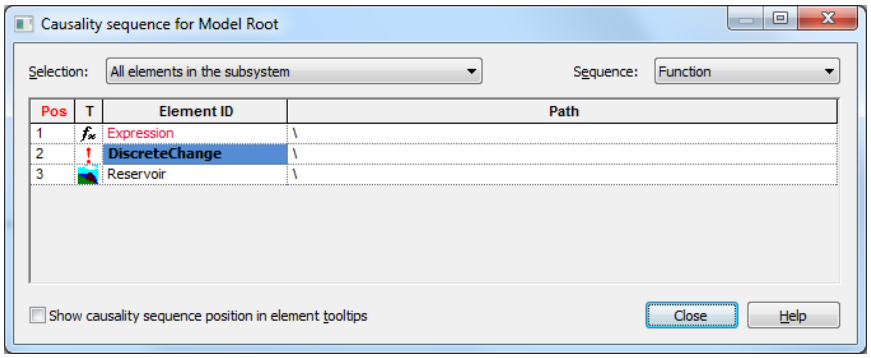
1. Compute the Expression after the Reservoir has been updated by the Discrete Change; or
2. Compute the Expression after the Reservoir has been brought up to the current time, but before it has been updated by the Discrete Change.

GoldSim will sequence it assuming the first option is correct. However, the second option is equally valid, and will produce a different result! Hence, when you have a system such as this, you must specifically instruct GoldSim how to sequence this system to ensure it accurately represents the system you are trying to model.

To facilitate this, GoldSim provides an option to manually force a particular causality sequence. If you right-click on an element in the sequence, one of the options is to "Specify Precedent". You can then specify a particular element that you wish the selected element to follow in the causality sequence. In this particular case, we want the Expression to be a precedent for the Discrete Change. Right-clicking on the Discrete Change and selecting "Specify Precedent" displays this dialog:



If we select Expression here, the causality sequence now looks like this:



Note that Expression is now in front of the Discrete Change. The Discrete Change is highlighted in blue (indicating that it has a "forced" precedence). The input port for the Discrete Change is also highlighted in blue, and its precedence requirement is added to its tool-tip:

Expression is shown in red in the causality sequence, indicating that this is not the default position for this element (GoldSim has forced it upward in the sequence).

If you right-click on the Discrete Change, you would see that there is a new option ("Remove Precedent").

Two other points regarding forced precedence should be noted:

- You can only specify one precedence requirement for an element (although an element can have multiple elements that use it as a precedent).

- GoldSim will not allow you to specify a precedent that results in an invalid causality sequence (e.g., a recursive system). It only allows you to force sequences that are still valid.

An example of the use of forced precedence to control sequencing (Bounce.gsm) can be found in the General Examples folder in your GoldSim directory.

Using SubModels to Embed Models Within Models

In some cases, you may want to embed an entire GoldSim model within another GoldSim model. GoldSim provides a special element called a SubModel to facilitate this.

A SubModel superficially looks like a Container, and conceptually shares some aspects with Containers. However, the functionality of a SubModel is quite different from a Container. Whereas a Container is simply a grouping of elements within a model, a SubModel is a completely separate "inner" model within an "outer" model. That is, it has its own simulation settings (time and Monte Carlo options) that are independent of the simulation settings of the outer GoldSim model within which the SubModel element is placed. Hence, when a SubModel element (i.e., the inner model) is triggered to do a calculation by the outer model, it runs a complete simulation.

The sections below provide the details on the use of SubModels in GoldSim.

What Can I Do With a SubModel?

Before describing the details of how SubModels are created and used, it is worthwhile to first discuss why you might want to use a SubModel.

Recall that a SubModel is a complete "inner" model that has been embedded in an "outer" model. The inner model and the outer model can take on a number of forms (e.g., static/dynamic, deterministic/Monte Carlo, simulation/optimization). This provides a wide variety of potential uses. The most common are summarized below.

Manipulation of Monte Carlo Statistics. In some cases, after carrying out a Monte Carlo simulation, you may want to carry out further calculations using the statistical outputs of the simulation. For example, you may want to carry out a calculation that is some function of the mean and the 95th percentile of a particular output in a Monte Carlo simulation. Without the use of SubModels, the only way to accomplish this is by exporting results from a Monte Carlo simulation manually to another application (e.g., a spreadsheet or a separate GoldSim model). With SubModels, this is easily accomplished within a single GoldSim model by inserting a SubModel (that performs a Monte Carlo simulation) into an outer model (that is static and simply manipulates the statistical outputs of the inner model).

Explicit Separation of Variability from Uncertainty. Many models have uncertain parameters as well as randomly variable parameters. An uncertain parameter represents ignorance that can theoretically (but perhaps not practically) be reduced through investigation (e.g., the mean failure time for a batch of light bulbs). Variability is inherent in many systems (e.g., the distribution of failure times for a batch of light bulbs) and cannot be reduced. It is often valuable to explicitly separate variability from uncertainty in a model. With SubModels, this is accomplished by inserting a SubModel (e.g., a dynamic Monte Carlo simulation) within an outer model (e.g., a static Monte Carlo simulation). This is referred to as "nested" Monte Carlo simulation. In the example above, the outer model would sample a probability distribution that represents the uncertainty in the mean

lifetime of a light bulb, and the inner model would simulate the performance of a number of random light bulbs (whose lifetime is sampled from a distribution with the mean specified by the outer model). The result of this type of analysis is a probability of probabilities.

Probabilistic Optimization. If you wish to optimize a probabilistic (uncertain) system, the objective function to be optimized cannot be a single deterministic output. Rather, it must be a statistic. That is, if X was an output of a probabilistic model (and hence was output as a probability distribution), optimizing X itself would be meaningless. Rather, you would need to optimize a particular statistic (e.g., the mean or 50th percentile) of the output X. With SubModels, this is accomplished by inserting a SubModel (e.g., a dynamic Monte Carlo simulation) within an outer model (e.g., a static optimization).

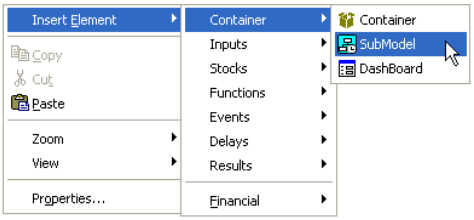
Dynamic Optimization During a Simulation. Imagine a situation where you were simulating the operation of a facility over a period of one year. Every month, the operators make a decision based on the current state of the system. This decision is based on a simple optimization analysis using currently available data (i.e., at every month during the simulation). The optimization chooses the optimum values of a few control variables that they will use for the next month. With SubModels, you could simulate this by inserting a SubModel (e.g., a static optimization) within an outer model (e.g., a dynamic simulation).

Example files describing all four of these applications can be found in the General Examples folder of your GoldSim directory (Submodel_1.gsm, Submodel_2.gsm, Submodel_3.gsm, and Submodel_4.gsm in the subfolder named SubModels).

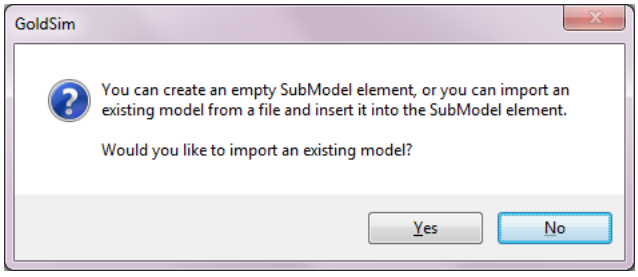
Read more: SubModel Examples (page 952).

Creating a SubModel

SubModels are listed under Container in the Insert Element context menu:



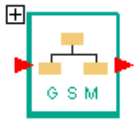
When you insert a SubModel, you are presented with a dialog asking if you want to create a new (empty) SubModel, or create the SubModel by importing an existing standalone model:



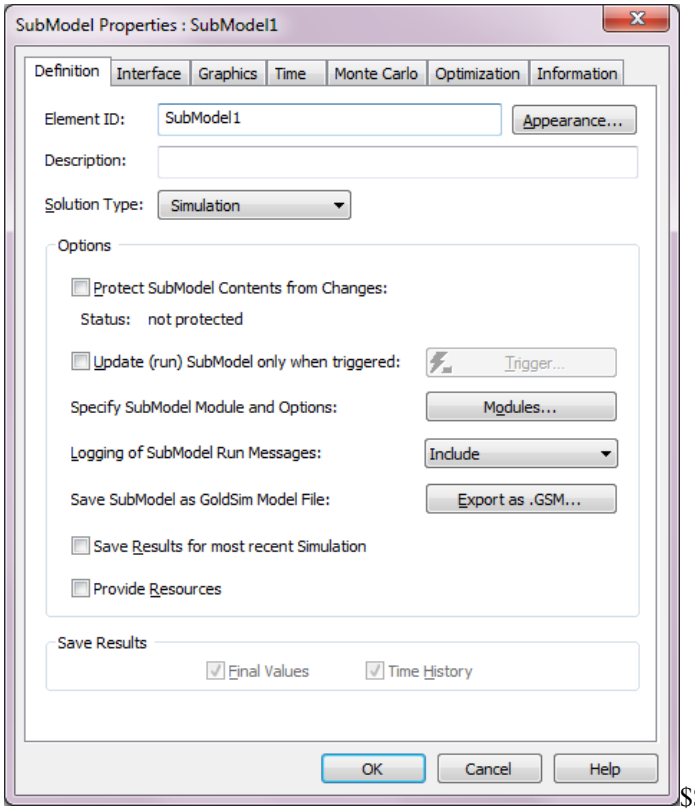
If you choose to import a standalone model, you will be prompted for the file name and a SubModel with the contents of that model will be inserted into GoldSim. Otherwise, a new (empty) SubModel will be created.

Read more: Importing SubModels (page 948).

The default icon for the SubModel element looks like this:



The property dialog looks like this:



$$$

Like all GoldSim elements, you first specify an **Element ID** and a **Description**.

After inserting the SubModel element, the steps for building a SubModel are generally as follows (although they will typically be done iteratively, rather than in this exact order):

1. Specify the Solution Type (Simulation or Optimization).
2. Specify (or if an existing model was imported, edit) the GoldSim modules required by the SubModel.
3. Build (or if an existing model was imported, edit) the contents of the SubModel.
4. Specify (or if an existing model was imported, edit) the simulation settings for the SubModel.
5. Create an input interface between the SubModel and the parent (outer) model.
6. Create an output interface between the SubModel and the parent (outer) model.
7. Specify when the SubModel is to be run.

These steps are described in detail in the sections below.

Specifying the Solution Type for a SubModel

The first step in creating a SubModel (after inserting the element) is to specify the **Solution Type**. There are two options: "Simulation" and "Optimization".

This option determines how the SubModel is being used. For most applications, you should choose "Simulation", which is the default.

"Optimization" will normally only be used when a SubModel is being used to carry out a dynamic optimization (i.e., at specified times) during a simulation. For example, imagine a situation where you were simulating the operation of a facility over a period of one year. Every month, the operators make a decision based on the current state of the system. This decision is based on a simple optimization analysis using currently available data (i.e., at every simulated month during the simulation). The optimization chooses the optimum values of a few control variables that they will use for the next month.

To represent this in GoldSim, you would need to specify the **Solution Type** for the SubModel as "Optimization", and the SubModel itself would represent the optimization calculations carried out by the simulated operator.

Note: If the Solution Type is "Optimization", Monte Carlo options for the SubModel must be set to "Deterministic".

Read more: Specifying the Simulation Settings for a SubModel (page 919).

If you do choose the Simulation Type as "Optimization", you will also need to specify the optimization settings in the **Optimization** tab for the SubModel.

Read more: Running an Optimization Within a SubModel (page 944).

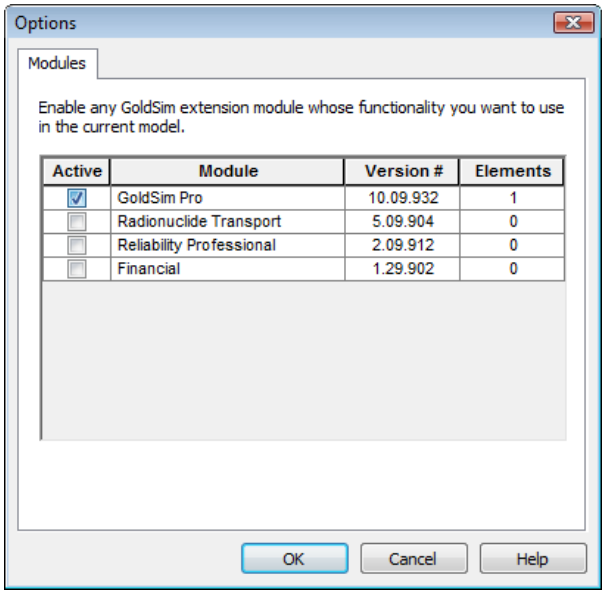
Specifying the Modules and Module Options for a SubModel

Because a SubModel is effectively a separate model (as opposed to a Container), you may choose to use different extension modules in a SubModel than in the outer model. For example, your SubModel may require the Contaminant Transport Module, while the outer model may not.

When you create a SubModel, the modules that are activated by default are the same as those that would be activated by default if you created a new model. (These defaults are set by the user by pressing the **Set as Default** button when activating modules in the **Modules** tab of the Options dialog).

Read more: Activating and Deactivating Extension Modules (page 26).

To specify the modules that will be activated for a SubModel, press the **Modules…** button on the SubModel dialog. The following dialog will be displayed:



All extension modules appear in the dialog. If your license does not allow you to use a particular module, it will be grayed out (and the **Active** column will be blank). You can activate and deactivate modules that you have permission to use by clicking the **Active** checkbox.

If you deactivate a module, the specialized elements associated with that module will be deleted from your SubModel (if any are present) and any associated menu options will be removed in the current file. If you make a module active, the various options associated with that module are made available again.

Note: If a module that is active in a SubModel has special options (that are normally accessed via a tab in the **Options** dialog for a standalone model), these options will appear in a tab on the Modules dialog for the SubModel.

Building the Contents of the SubModel

Once you have selected the Solution Type and modules to be used by a SubModel, building the contents of a SubModel is, for the most part, identical to building the contents of a Container.

Like a Container, you enter a SubModel by clicking on the plus sign in the upper left hand corner of the element. SubModels (and their contents) are also displayed in the browser in the same manner as Containers.

You can add elements, Containers, and even other SubModels to a SubModel, just as you would to a Container. Of course, you can also copy elements that are outside of a SubModel and paste them into the SubModel.

Note: In addition to creating a model manually inside a SubModel, you can also choose to import an existing (standalone) model into a SubModel.

Read more: Importing SubModels (page 948).

There are several important points to note when building and navigating a SubModel:

- When you are inside a SubModel, the navigation bar at the top of the graphics pane only shows the location with respect to the SubModel. It does not show the path relative to the parent model:

  Note also that the background in the field where the path is displayed is a different color. This is intended to provide a visual indication that you are inside a SubModel.

- Within a SubModel, you cannot reference outputs that exist outside of the SubModel in the same way you would from inside a Container. That is, by default, the SubModel is a self-contained system (i.e., a separate model) that cannot "see" anything on the outside. In fact, within a SubModel, the Insert Link dialog accessed via the context menu for input fields does not list any elements outside of the SubModel. In order to access outputs from outside the SubModel, you need to take some specific actions (i.e., add the output to the SubModel's input interface).

  Read more: Creating the Input Interface to a SubModel (page 921).

- Because the SubModel is a self-contained system (i.e., a separate model), elements on the outside cannot directly reference outputs inside the SubModel. Of course, in order to be of any value, a SubModel must have some way to communicate with (i.e., provide outputs to) the outer model. However, in order to provide outputs from inside the SubModel to elements in the outer model, you need to take some specific actions (i.e., add the outputs to the SubModel's output interface), and the SubModel outputs themselves have added complexity (since they may represent complex results, such as a distribution resulting from a Monte Carlo simulation).

  Read more: Creating the Output Interface to a SubModel (page 925).

- Results inside a SubModel behave differently than results that have been added to the SubModel's output interface. In particular, because a SubModel is a separate simulation (that will typically be carried out multiple times during a simulation of the outer model), results *inside* of a SubModel are typically overwritten during a simulation of the outer model. As a result, although you can choose to save the results inside of a SubModel, only results from the *last* simulation of the SubModel (i.e., the last time the SubModel was run) are available for viewing *inside* the SubModel at the end of your simulation. Note, however, that this does not apply to results that have been added to the SubModel's output interface. A wide variety of complex results can be accessed via the interface, including various statistics and raw data for *all* SubModel time histories for all simulations of the SubModel.

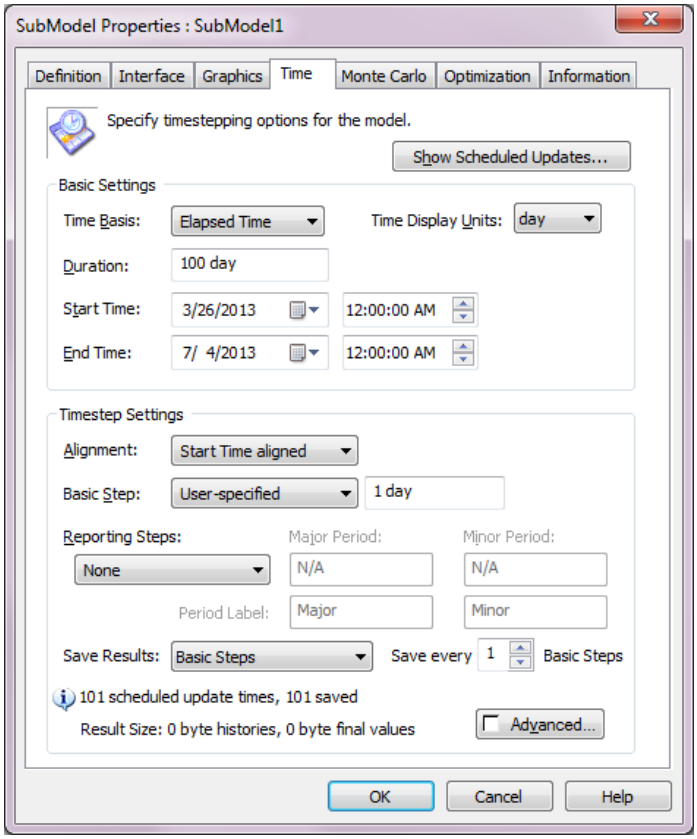
  Read more: Saving and Viewing Results Inside a SubModel (page 932); Creating the Output Interface to a SubModel (page 925).

Specifying the Simulation Settings for a SubModel

Because a SubModel is a separate model, it requires its own simulation settings (i.e., timestepping and Monte Carlo options). That is, a SubModel does not use the simulation settings specified by the outer model.

The simulation settings for a SubModel are accessed via tabs at the top of the SubModel dialog.

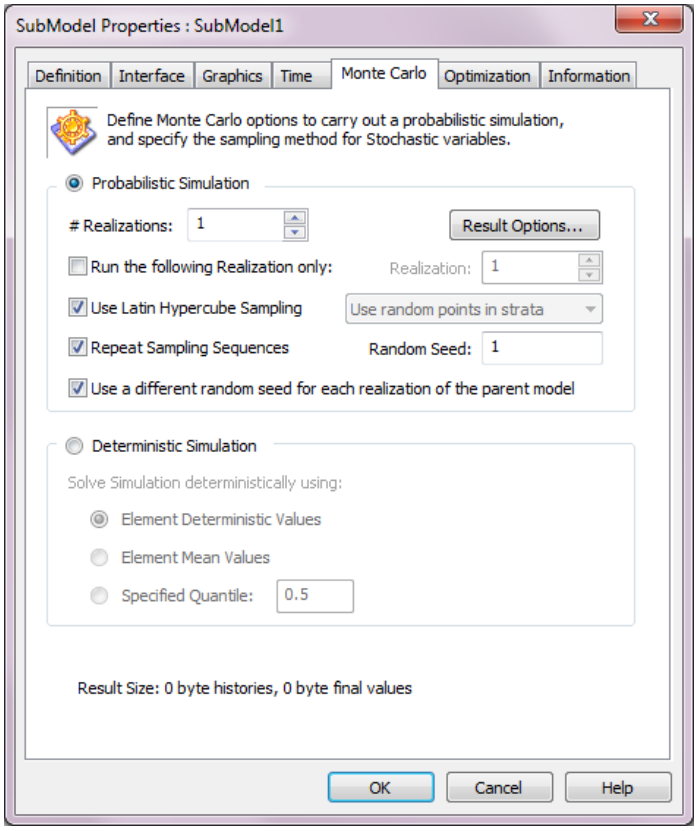
The **Time** tab is used to specify the time options for the simulation:



The inputs to be entered here are identical to those provided in the **Time** tab of the Simulation Settings dialog for a model.

Read more: Setting the Basic Time Options (page 413).

The **Monte Carlo** tab is used to specify the Monte Carlo options for a simulation:



With one exception, the inputs to be entered here are identical to those provided in the **Monte Carlo** tab of the Simulation Settings dialog for a model.

Read more: Setting the Monte Carlo Options (page 438).

The one exception is the **Use a different random number seed for each realization of the parent model** field. If this option is checked (the default), the Monte Carlo SubModel will behave randomly, producing a different result for each realization of the parent model and for each time the SubModel is calculated. If this box is cleared, the Monte Carlo SubModel will behave identically for each realization of the parent model, so that, for example, realization 17 of the inner model is the same for every realization of the parent model.

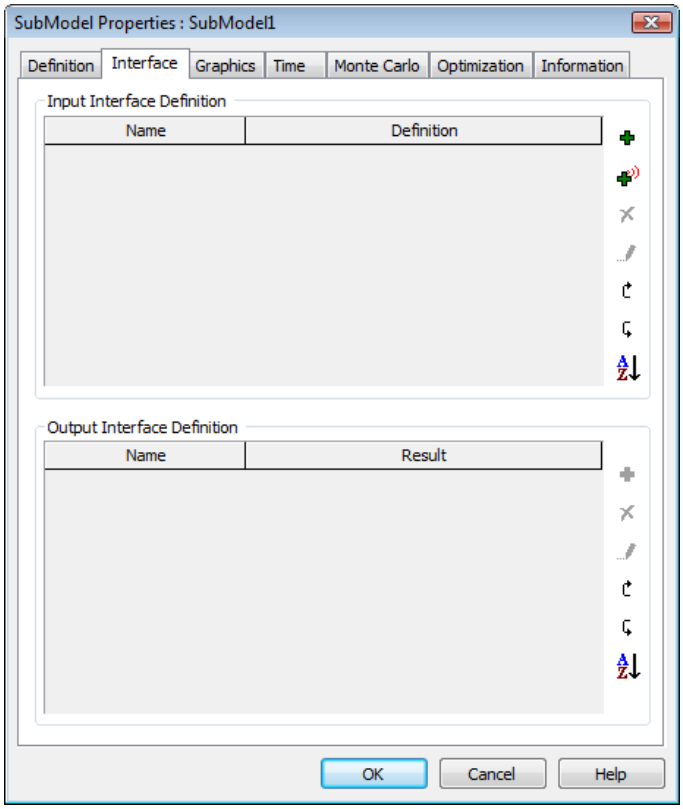
Note: You can choose to specify some of the simulation settings for a SubModel via the Input Interface for the SubModel. If you do this, some of the input fields in the **Time** and **Monte Carlo** tabs will become locked.

Creating the Input Interface to a SubModel

Because a SubModel is a self-contained system (i.e., a separate model), elements inside a SubModel cannot "see" anything on the outside (i.e., in the outer model). As a result, you cannot reference outputs that exist outside of the SubModel in the same way you would from inside a Container. (In fact, within a SubModel, the Insert Link dialog accessed via the context menu for input fields does not list any elements outside of the SubModel).

Of course, in order to be of any value, a SubModel must have some way to communicate with (e.g., access outputs of) the outer model. This is done by creating an interface between the SubModel and the outer model.

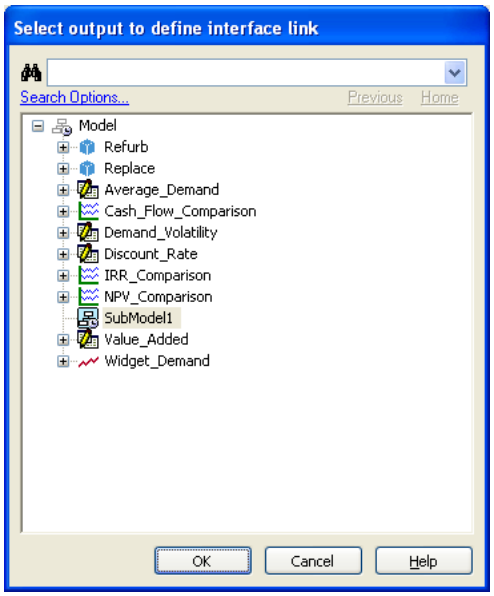
The interface is accessed via the **Interface** tab on the SubModel dialog:



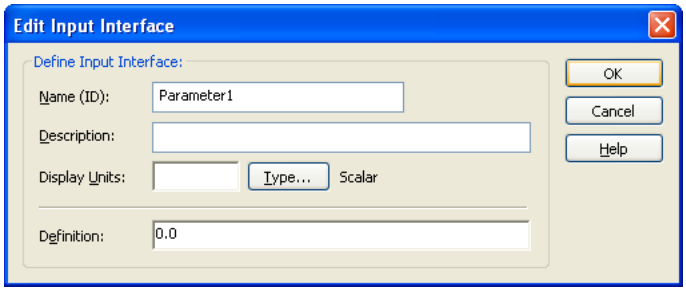
The top part of this dialog is used to define the input interface. The bottom portion is used to define the output interface.

Add button

In order to access an output outside of the SubModel, you must add the output to the input interface. This is done by pressing the Add button in the Input Interface portion of the dialog. When you do this, GoldSim displays the Insert Link dialog, allowing you to select an output outside of the SubModel to add to the interface):



After selecting a link and pressing **OK** (or selecting **Cancel**), the following dialog is displayed:



The **Name (ID)** is the name by which the output variable can be referenced inside the SubModel. This has the same restrictions as an element ID. The **Description** is an optional description of the variable.

You must then specify the **Display Units** and **Type** information (e.g., value/condition, scalar/vector/matrix) for the variable. Finally, you specify the **Definition** for the input. Typically, this will be a link to an output in the outer model. However, it can also be an equation (with links), or a constant value.

Note: If you selected a link (rather than pressing **Cancel**) when adding the interface input, the Definition, Display Units and Type will automatically be entered.

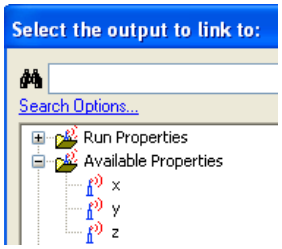
In addition to Values and Conditions, three types of complex outputs can be specified as inputs to a SubModel Input inteface: Time Series Definitions, Lookup Table definitions, and Distribution definitions. These are specified via the **Type** button. In the two former cases, additional information (e.g., what the Time Series represents, units for the independent variables for a Lookup Table) must also be specified in the input interface dialog.

A Time Series definition can be produced by a Time Series element, an External element or another SubModel. A Lookup Table definition can be produced by an External element. A Distribution definition is one of the outputs of a Stochastic element (and can also be produced by another SubModel). Passing these definitions through the input interface allows you to pass complete time series, lookup table and distribution definitions into the SubModel, where they can be used to define Time Series, Lookup Table or Stochastic elements.

Read more: Referencing a Time Series Definition Output (page 226); Using an External Element to Define Lookup Tables (page 880); Externally-Defined Distribution (page 168).

Once you have added an output from the outer model to the Input Interface of a SubModel, you can subsequently reference the output within the SubModel by using the specified **Name** that you defined previously, preceded by a ~. For example, if you create an interface input called X, you could reference it in input fields within the SubModel as ~X.

You can also use the Insert Link option to reference an item on the Input Interface by right-clicking in an input field within the SubModel. When you do so, all of the items on the Input Interface to the SubModel will be displayed within the "Available Properties" folder:



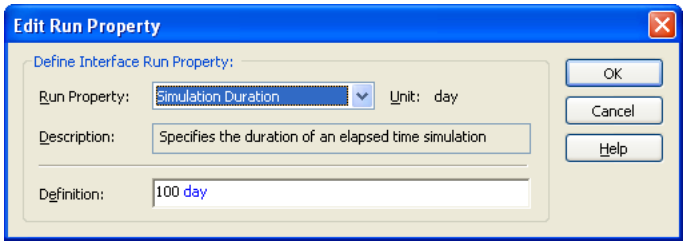
Note: In this case, the ~ indicates that this output has some special characteristics (in particular, it is coming from outside of the SubModel). This is an instance of what is referred to as a locally available property in GoldSim. Locally available properties are accessed by using a ~.

Read more: Understanding Locally Available Properties (page 750).

In some cases, you may want to control some of the simulation settings (i.e., time and Monte Carlo options) of the SubModel from outside of the SubModel (e.g., using values provided by the outer model). To facilitate this, GoldSim allows you to add run properties to the input interface.

Add Run Property button

This is done by pressing the Add Run Property button in the Input Interface portion of the dialog. When you do this, the following dialog is displayed:



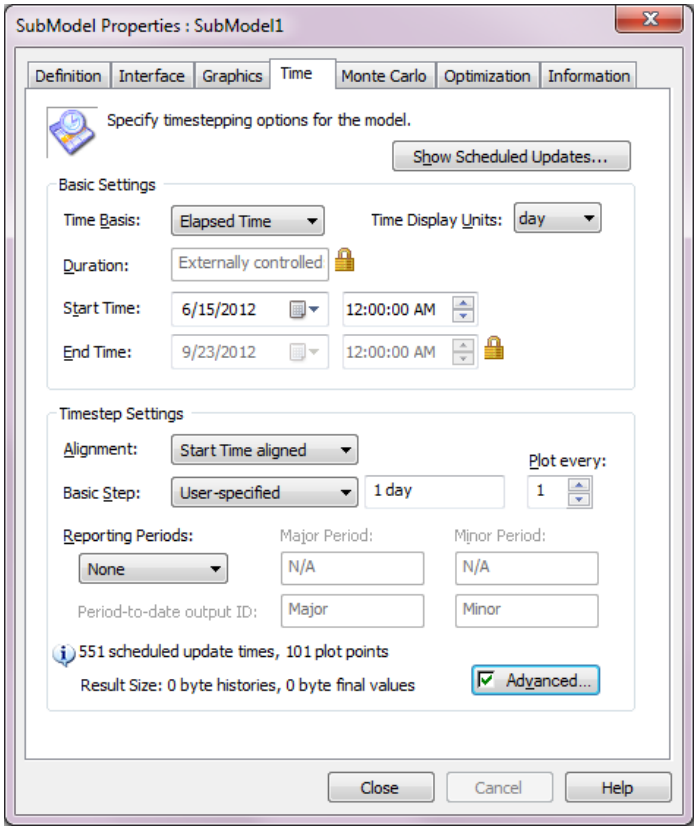
The Run Property field provides a drop-list of options for the SubModel simulation setting that you wish to control. The options are:

- Simulation Duration;
- Maximum time between updates;
- Number of Realizations; and
- Realization to Run

The **Definition** field determines the value that will be used for the selected simulation setting input. Typically, this will be a link to an output in the outer model (the **Definition** field's context menu can be used to access an Insert Link dialog that lists all elements in the outer model). However, it can also be an equation (with links), or a constant value.

Note: If you wish to control the Number of Realizations Run Property, the value must always be set to a number greater than 1. That is, you cannot switch between a single realization run and a multi-realization run of the SubModel dynamically via the Input Interface.

Note that when you select one of these options, the appropriate input fields are automatically locked (and cannot be edited) in the **Time** and **Monte Carlo** tabs for the SubModel:



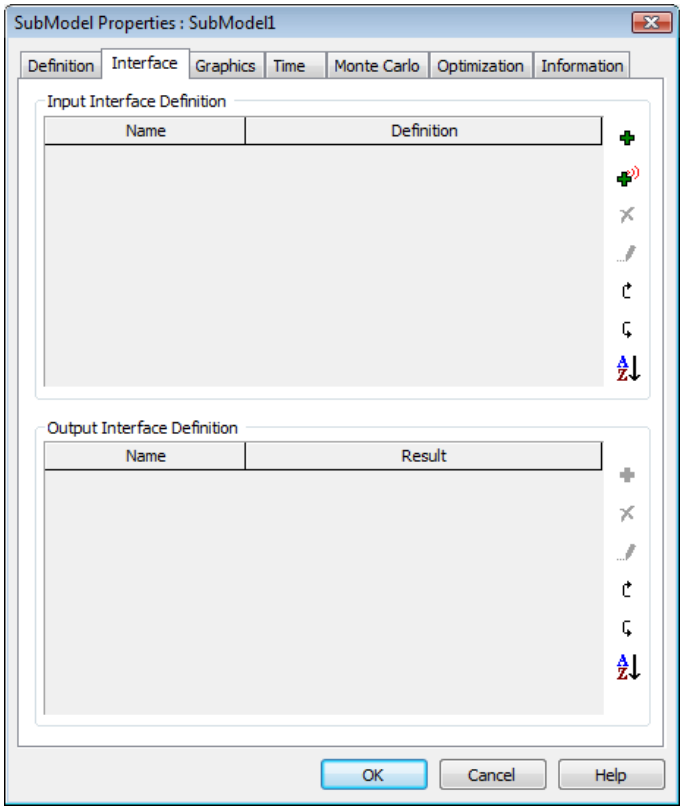
Note: Buttons for deleting, editing, moving and alphabetically sorting the inputs on the interface are available directly below the Add and Add Run Property buttons.

Creating the Output Interface to a SubModel

Because a SubModel is a self-contained system (i.e., a separate model), by default, elements outside a SubModel cannot "see" anything on the inside. Moreover, because a SubModel is a separate simulation (that will often be carried out multiple times during a simulation of the outer model), intermediate results of a SubModel are typically overwritten during a simulation of the outer model and are not available in the outer model.

Of course, in order to be of any value, a SubModel must have some way to communicate with (e.g., provide outputs to) the outer model. This is done by creating an output interface between the SubModel and the outer model.

The output interface is accessed via the **Interface** tab on the SubModel dialog:



The top part of this dialog is used to define the input interface. The bottom portion is used to define the output interface.

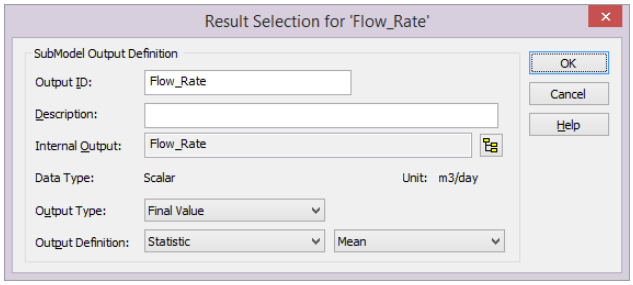
Add button

The output interface allows you to specify the outputs that you want the SubModel to provide to the outer model. It is important to understand that the output interface represents the only mechanism by which you can access outputs of a SubModel outside of the SubModel.

In order to allow the outer model to access an output of the SubModel, you must select the output and add it to the output interface. This is done by pressing the Add button in the Output Interface portion of the dialog.

When you press this button, you will be presented with a browser displaying the outputs inside the SubModel. You must select the output that you are interested in.

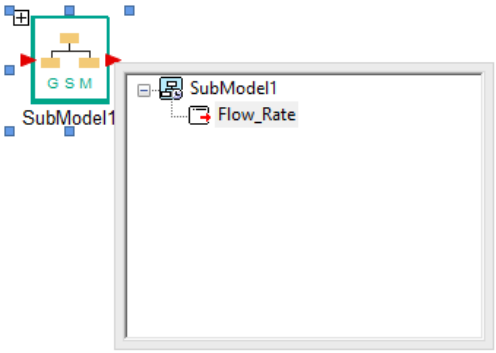
After you select an output, the following dialog will be displayed:



The **Output ID** is the name by which the output variable can be referenced in the outer model. This ID has the same restrictions as an element ID.

Note that the outputs defined in the output interface are added to the output port interface of the SubModel and can be accessed there (in exactly the same

manner the outputs from other kinds of elements are accessed). The Output ID is what is shown in the interface:



The **Description** is an optional description of the output. The **Internal Output** field displays the selected output (which you can change using the button to the right of the field).

The **Data Type** and **Unit** fields are automatically populated based on the **Internal Output** selection.

After selecting the **Internal Output** you must then select what type of result is to be made available to the parent model on the output interface. This is controlled by two fields: **Output Type** and **Output Definition**. For some **Output Definition** selections, a third field will be presented to the immediate right of the **Output Definition** field.

There are two options for **Output Type**:

- Final Value. The following **Output Definitions** are available (some of which are only valid selections under specific circumstances):
  - Last Calculated. This selection is always valid.
  - Statistic. This selection is only valid for Probabilistic SubModels (it is not valid for Deterministic SubModels). A specific Statistic must be defined immediately to the right of the **Output Definition** field.
  - Distribution. This selection is only valid for Probabablistic SubModels and scalar outputs (it is not valid for Deterministic SubModels or for array outputs).
- Time History. The following **Output Definitions** are available:
  - Last Calculated. This selection is only valid for dynamic SubModels and scalar outputs (it is not valid for Static SubModels or for array outputs).
  - Statistic History. This selection is only valid for Probabilistic, dynamic SubModels and scalar outputs (it is not valid for Deterministic or Static SubModels or for array outputs). A specific Statistic must be defined immediately to the right of the **Output Definition** field.
  - Realization Histories. This selection is only valid for Probabilistic, dynamic SubModels and scalar outputs (it is not valid for Deterministic or Static SubModels or for array outputs).

Note: For some advanced applications, you may select a Time Series Definition output (from an External element or Time Series element inside the SubModel) rather than a Value as the **Internal Output**. If you do so, the **Output Type** is automatically set to be a Time Series Definition (rather than a Final Value or Time History).

When you add an output to the Output Interface and select the **Output Type** and **Output Definition**, you will be notified if your selections are invalid.

The Output Interface provides a summary of all of the outputs (as well as their **Output Type** and **Output Definition)**:

Output Interface Definition

Name	Result
Final_Volume	Final Value
Mean_Volume	Statistic (Mean)
Final_Volume_Distribution	Final Value Distribution
Rate_Final_Realization	Time History
Mean_Flow_Rate	Statistic History (Mean)
Rate_All_Realizations	All Realization Histories

If you change the settings such that some of the selections are no longer valid, the Results will be identified in red (and the model will not run). In the example below, the SubModel has been set to Determinstic, invalidating those Output Types that require a Probabilistic SubModel:

Output Interface Definition

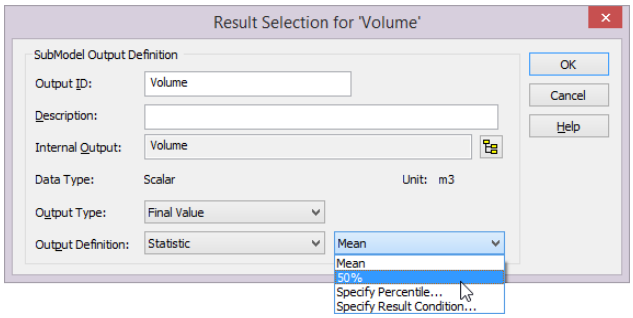
Name	Result
Final_Volume	Final Value
Mean_Volume	*Statistic (Mean)*
Final_Volume_Distribution	*Final Value Distribution*
Rate_Final_Realization	Time History
Mean_Flow_Rate	*Statistic History (Mean)*
Rate_All_Realizations	*All Realization Histories*

It is important to understand what these various results are and how they can be used in the parent model:

Final Value Results

Last Calculated. This result is simply the final value of the output that was computed at the end of the simulation (for the final realization of the SubModel if multiple realizations were run). This will be listed in the Output Interface as "Final Value". It can be used directly in the parent model as any other value (or condition) output could be used.

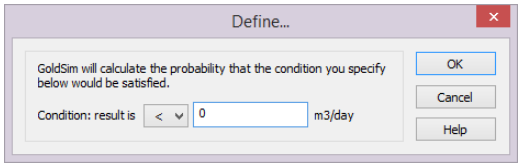
Statistic. If this is selected as the Output Definition, a third field appears directly to the right:



The first two options are "Mean" and "50%" (the Median). The third option provides access to a dialog where you can specify any percentile. In these three

cases, the result represents the specified statistic (computed over all realizations of the SubModel) for the final value of the selected output (i.e., the value computed at the end of each realization).

The fourth option ("Specify Result Condition") provides access to the following dialog:



In this case, the result represents the probability (computed over all realizations of the SubModel) that the final value of the output satisfies the specified condition. In the example above, within the parent model, the result would represent the probability that the final value of the selected SubModel output was less than 0 m3/day.

In all cases, the result will be listed in the Output Interface as "Final Value (*Statistic*)", where *Statistic* is the selected option. It can be used directly in the parent model as any other value (or condition) output could be used.

Distribution. This result is a complex output (referred to as a Distribution output) that represent all the statistical information necessary to define a probability distribution computed by the SubModel. It can be used to define a Stochastic in the parent model (as an "externally-defined" distribution). More frequently, it will serve as an input to specialized functions that operate on distributions. For example, PDF_Mean(X) returns the mean of the distribution X; PDF_Value(X,0.95) returns the 95% percentile. Under certain conditions (nested Monte Carlo simulation), Final Value Distributions can represent a distribution of distributions.

The result will be listed in the Output Interface as "Final Value Distribution". Unlike other results, because it is a complex type of output, it can only be used in the parent model in specific locations (e.g., as input to an externally-defined Stochastic, as input to a specialized distribution function, within a Distribution Result element).

Read more: Externally-Defined Distribution (page 168); Specialized Functions That Operate on Distributions (page 183); Adding a Distribution Output to a Distribution Result (page 620); Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933).

Time History Results

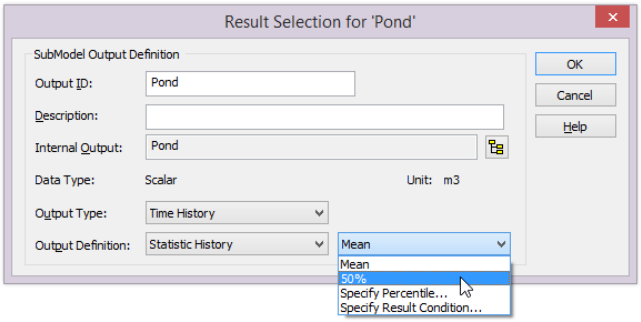
Time History results on a SubModel Output Interface are complex ouputs that represent all the information necessary to define a time series. As such, they can only be used in three places within the parent model: 1) to define a Time Series element in the parent model; 2) as an input to an External (DLL) element; or 3) directly within a Time History Result.

Read more: Referencing a Time Series Definition Output (page 226); Using an External Element to Read and/or Output Time Series (page 881); Viewing SubModel Results in Time History Result Elements (page 573).

Last Calculated. This result is the time history of the output for the final realization of the SubModel (which is equivalent to the *only* realization if the SubModel Monte Carlo Settings do not specify multiple realizations). Hence, regardless of whether the SubModel is probabilistic or deterministic, it always

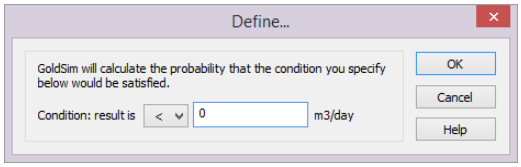
consists of a single time history (for each realization of the parent model). This will be listed in the Output Interface as "Time History".

Statistic History. If this is selected as the Output Definition, a third field appears directly to the right:



The first two options are "Mean" and "50%" (the Median). The third option provides access to a dialog where you can specify any percentile. In these three cases, the result represents a time history of the specified statistic (computed over all realizations of the SubModel) for the selected output.

The fourth option ("Specify Result Condition") provides access to the following dialog:



In this case, the result represents a time history of the probability (computed over all realizations of the SubModel) that the output satisfies the specified condition. In the example above, within the parent model, the result would represent the time history of the probability that the selected SubModel output was less than 0 m3/day.

In all cases, regardless of whether the SubModel is probabilistic or deterministic, it always consists of a single time history (for each realization of the parent model). This will be listed in the Output Interface as "Statistic History (*Statistic*)", where *Statistic* is the selected option.

Realization Histories. This result is the time history of the output for *all* realizations of the SubModel. Hence, unless the SubModel Monte Carlo Settings do not specify multiple realizations, it consists of multiple time histores (for each realization of the parent model). Hence, when running nested Monte Carlo simulation (i.e., in which both the parent model and the SubModel are probabilitic), this can result in a very complex set of results. For example, if the parent model was run for 100 realizations, and the SubModel was run for 50 realizations, the results being displayed in a Time History Result element within the parent model would be based on 5000 individual time histories (50 * 100). Fortunately, GoldSim provides some powerful tools to view this complex set of results.

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933); Viewing SubModel Results in Time History Result Elements (page 573).

This type of result will be listed in the Output Interface as "All Realization Histories".

Note: Calculations in GoldSim are carried out in double-precision. However, outputs of a SubModel are reduced to single precision when they pass through the output interface.

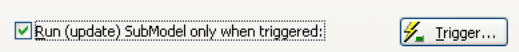
Note: Buttons for deleting, editing, moving and alphabetically sorting the outputs on the interface are available directly below the Add button.

At the bottom of the Definition tab for SubModels are options to Save Results for **Final Values** and **Time History**. The outputs for a SubModel, of course, are those results that have been added to the output interface. However, it is important to understand that these flags do not apply to *all* outputs in the output interface. In particular, they only apply to "standard" outputs (which must be Last Calculated or Statistic Final Values). They do not apply to the special output types discussed above (Distribution outputs or Time Series Defintion outputs). These output types can only be viewed through Result elements in the parent model. (Note that time histories of "standard" SubModel results as viewed in a dynamic parent model would only change if the SubModel was called multiple times during simulation of the parent model).

Controlling When a SubModel is to be Run

By default, a SubModel is treated like any other element in GoldSim and is updated (i.e., run) whenever required to do so by the parent model (e.g., when inputs change).

In some cases, however, you may want to manually control when a SubModel is updated. You can do so by checking the **Run (update) SubModel only when triggered** checkbox. When you do this, access is provided to a standard trigger dialog:



You can explicitly control when a SubModel is run by defining one or more triggers (e.g., "On True: X > Y").

Read more: Understanding Event Triggering (page 323).

By default, when this box is checked, no triggers are defined. This means that the SubModel will never run.

Note: You can tell if a trigger has been defined from the appearance of the **Trigger…** button. If a trigger is defined, the rectangle next to the lightning bolt is bright green; otherwise it is dark green. And like all **Trigger…** buttons, it displays a tool-tip. If there is no trigger defined, the tool-tip will display "Never updates SubModel".

If you wish the SubModel to run when its parent Container activates, set the update trigger to **Auto Trigger**. Note that if a SubModel is set to **Auto Trigger** and it is not within a conditional Container, it activates when the Model (root) Container activates (i.e., at the start of the realization).

Prior to a SubModel being triggered, its outputs are all zero. The outputs are updated when the SubModel is triggered, and outputs remain constant until the SubModel is triggered again.

Other SubModel Options

Saving and Viewing Results Inside a SubModel

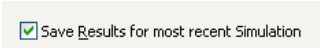
Several advanced options are available for SubModels.

These are discussed in the sections below.

Because a SubModel is a separate simulation (that will often be carried out multiple times during a simulation of the parent model), results *inside* of a SubModel are typically overwritten during a simulation of the parent model.

Accordingly, by default, other than results passed through the SubModel's output interface, no results are actually saved (and hence, you cannot view results inside a SubModel after running a simulation).

However, you can choose to save (and view) the results of the *last simulation* of the SubModel. This can be done by checking the **Save Results for most recent Simulation** checkbox on the SubModel's **Definition** tab:



If this box is checked, results from the *last* simulation of the SubModel (i.e., the last time the SubModel was run) are available for viewing inside the SubModel at the end of your simulation. Note that the SubModel's results could represent multiple realizations, and in such a case, both time history and probabilistic results would be available for viewing inside of the SubModel. However, if the SubModel was run multiple times by the outer model (e.g., every timestep of the outer model), only the results from the last time the SubModel was run would be available.

Note: There are two important exceptions to this. In particular, under certain circumstances, Result elements in the parent model can display multiple realizations from the SubModel: 1) If you are running a nested Monte Carlo simulation (both the SubModel and the parent model are running multiple realizations), Distribution Result elements in the parent model modify their behavior to enable you to view multiple sets of realizations produced by the SubModel. 2) You can select a special option in Time History Result elements in the parent model to enable you to view multiple time histories from within a SubModel.

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933); Viewing SubModel Results in Time History Result Elements (page 573).

Several points should be noted regarding saving and viewing results inside a SubModel:

- When carrying out distributed processing simulations (using the Distributed Processing Module), results *are not saved* inside a SubModel, even if you choose to **Save Results for most recent Simulation**.

Note: The Distributed Processing Module is discussed in detail in the **GoldSim Distributed Processing Module User's Guide**.

- If the **Save Results for most recent Simulation** box is cleared, results will *not be saved* inside the SubModel, even if the SubModel contains Result elements and/or the Save Results options are selected for each

element. (If you click on a Result element in Result Mode inside such a SubModel, it will be ignored).

- If 1) you have chosen to **Save Results for most recent Simulation**, 2) have added a Time History Result element; and 3) the element is specified to automatically export results to a spreadsheet or text file, GoldSim will not automatically export the results. If you wish to export the results of the last simulation from inside a SubModel, it can only be done manually.

Read more: Exporting from a Time History Result Element to a Spreadsheet (page 678); Exporting from a Time History Result Element to a Text File (page 686).

Carrying Out Nested Monte Carlo Simulation Using a SubModel

Many models have uncertain (epistemic) parameters as well as randomly variable (aleatory) parameters. An uncertain parameter represents ignorance that can theoretically (but perhaps not practically) be reduced through investigation (e.g., the mean failure time for a batch of light bulbs). Variability is inherent in many systems (e.g., the distribution of failure times for a batch of light bulbs) and cannot be reduced. It is often valuable to explicitly separate variability from uncertainty in a model.

With SubModels, this is accomplished by inserting a SubModel (e.g., a dynamic Monte Carlo simulation) within an outer model (e.g., a static Monte Carlo simulation). This is referred to as "nested" Monte Carlo simulation. The inner model simulates the random variability of the system, while the outer model represents the ignorance in key parameters.

In the example above, the outer model would sample a probability distribution that represents the uncertainty in the mean lifetime of a light bulb, and the inner model would simulate the performance of a number of random light bulbs (whose lifetime is sampled from a distribution with the mean specified by the outer model).

Note that for this kind of model, if you were to link an output from the SubModel to the Output Interface as s distribution, conceptually on the outside of the SubModel the output would actually represent a *distribution of distributions*. That is, because the SubModel is a Monte Carlo model, every time it is called, it produces a distribution. However, because the outer model itself is a Monte Carlo simulation, every element inside it results in a distribution of results. Hence, running the outer model multiple times results in a distribution of distributions for any SubModel output.

If you were to go inside the SubModel and view a result, it would display only the results from the last time the SubModel was run (and only if **Save Results for most recent Simulation** was checked).

However, on the outside of the SubModel, for a nested Monte Carlo run, it is necessary to display the *distribution of distributions*. Of course, GoldSim's standard Distribution Result only displays a single distribution. (It can display distributions from multiple outputs, but this is not the same as displaying nested distributions).

Read more: Viewing Distribution Results (page 596).

To support nested Monte Carlo simulations, Distribution Result elements modify their behavior under certain circumstances to enable display of nested Monte Carlo results. In particular, Distribution Result elements modify their behavior if and only if the following conditions are all met:

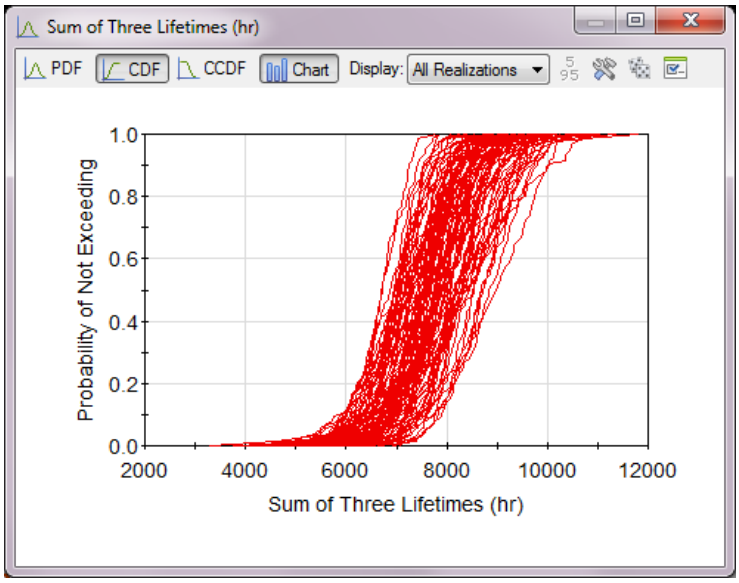
- The Distribution Result element is located in the outer model (i.e., not in a SubModel);
- A single result is selected for display in the Result element;
- The selected result references a distribution type output from a Monte Carlo SubModel (for this to occur, the SubModel must be configured to run multiple realizations);
- The result was added to the Result element prior to running the model;
- The SubModel whose result is being referenced is located within the main model (i.e., it cannot be nested inside another SubModel); and
- The outer model is configured to run multiple realizations.

Under these circumstances, there are two differences from a standard Distribution display:

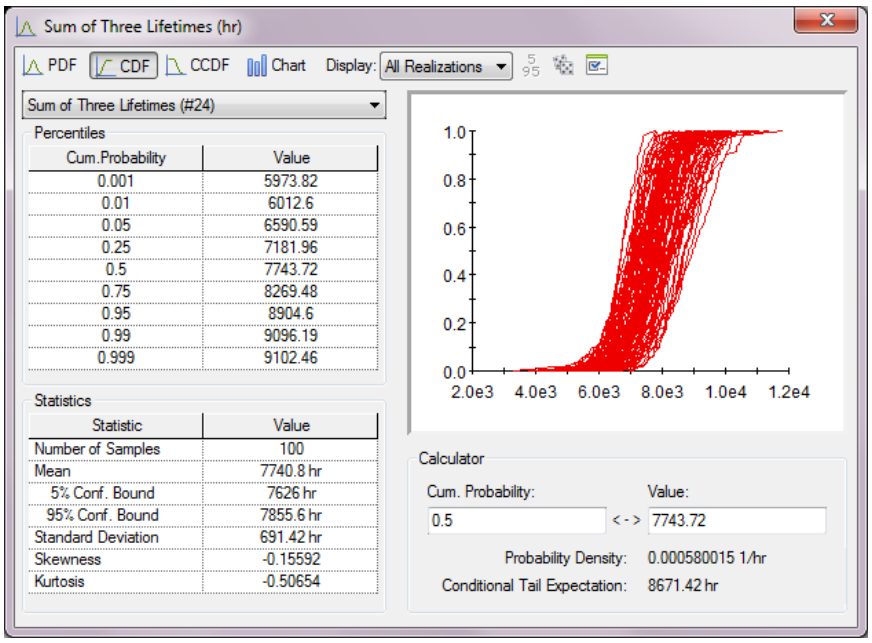
1. There is no option to display a Table; and
2. A **Display** option is provided. The options are:
   - All Realizations
   - Realization
   - Statistics
   - Probability
   - All Results

The various displays (and what they represent) are as follows:

All Realizations. "Realizations" refers to realizations of the outer model. Since each realization of the outer model produces a distribution, showing "All Realizations" simply displays multiple distributions (one for each realization of the outer model). The chart display looks like this:

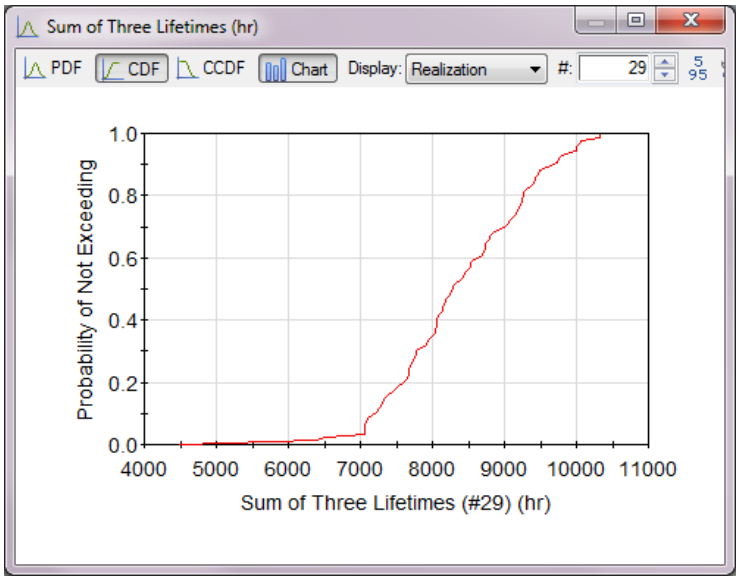


The Distribution Summary view looks like this:

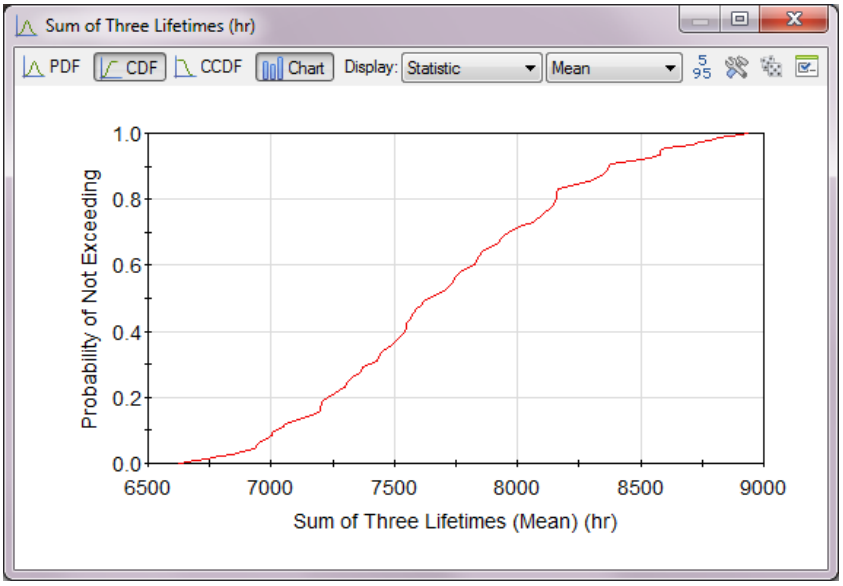


Statistics can only be shown for one realization at a time (which you can select from a drop list at the top of the dialog). In the example above, statistics are being shown for realization #24 of the outer model.

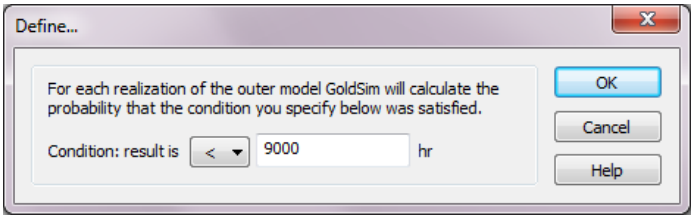
Realization. "Realization" refers to the realization of the outer model (which you must select). The chart (and Distribution Summary) then show that single outer model realization (i.e., a single distribution):



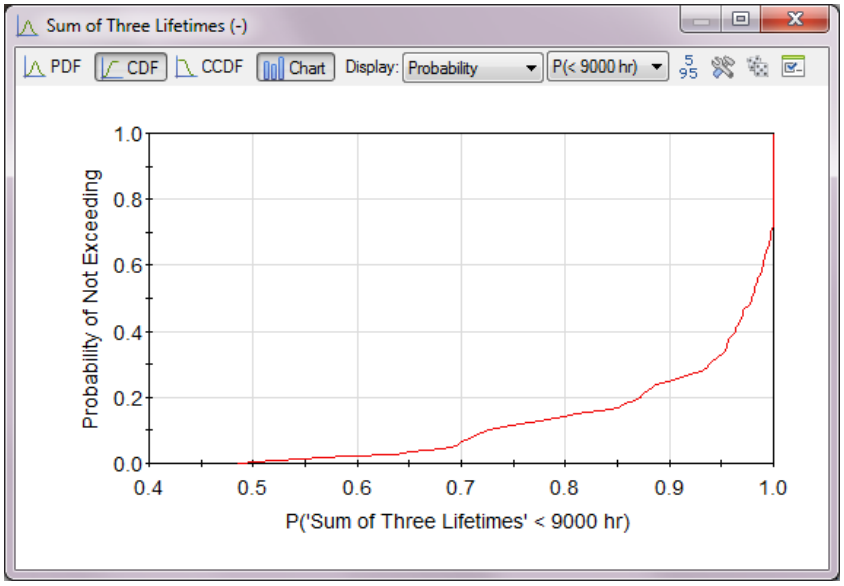
Statistic. When you select this option, a drop-list to the immediate right allows you to select or define a statistic. The result displayed is then the distribution of the specified statistic (each outer model realization produces one value for this statistic):



Probability. This option displays a probability distribution of probabilities. When you select this option, a drop-list to the immediate right allows you to define the probability that you are interested in, based on a condition:



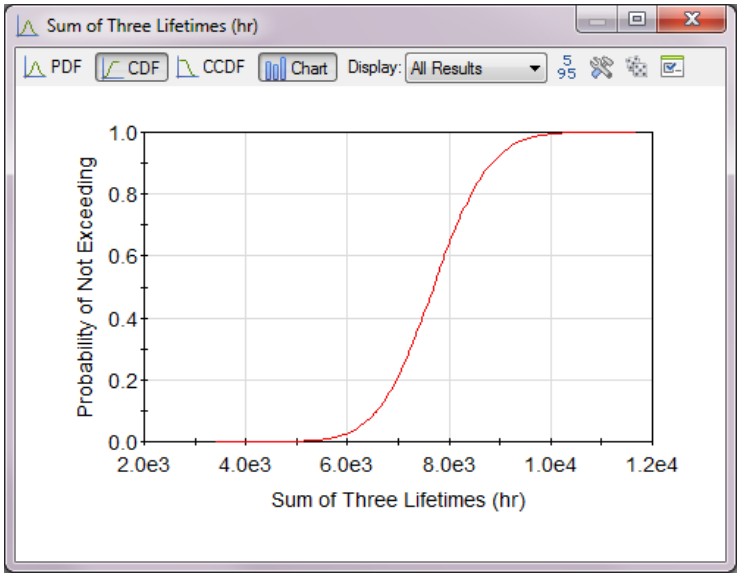
This instructs GoldSim to compute the (inner model) probability of a particular outcome for the specified result. Each outer model realization produces one value for this probability. It is the probability of the inner model's result being <, <=, =, >= or > the specified value. In the example above, for each outer model realization, GoldSim computes a single value: the probability of the output being less than 9000 hours. The overall result then displays a distribution that quantifies the uncertainty in this particular probability:



How do we interpret such a distribution? This is meant to represent a complex concept (a probability distribution of a probability!). Perhaps this is best understood by examining this particular example. In this example, we could state the following:

- There is approximately a 70% chance that the probability of the result being less than 9000 hours will be less than one. That is, in 70% of the outer model realizations, the probability of the result being less than 9000 hours was less than one, and in 30% of the realizations it was exactly one.
- There is no chance that the probability of the result being less than 9000 hours is below (approximately) 48%. That is, in none of the outer model realizations was the probability of the result being less than 9000 hours less than (approximately) 48%.

All Results. This combines all realizations together to form a single distribution that represents both variability (from the inner SubModel) and uncertainty (from the outer model). Hence, if you run 50 realizations of the SubModel and 100 realizations of the outer model, the distribution would be constructed from 5000 realizations.The chart (and Distribution Summary) then show the single distribution that represents all model uncertainties:



The text above describes how *distribution results* can be viewed for a nested Monte Carlo simulation. But what if you wanted to view time history results? Viewing time history results for nested Monte Carlo simulations is complicated by the fact that Time History Definition SubModel outputs consist of *multiple* time histories for each realization of the parent model. As a result, if the parent model was run for 100 realizations, and the SubModel was run for 50 realizations, the corresponding time history results would be based on 5000 individual time histories (50 * 100).

Read more: Viewing SubModel Results in Time History Result Elements (page 573).

To support nested Monte Carlo simulations, Time History Result elements modify their behavior under certain circumstances to enable display of nested Monte Carlo results. In particular, Time History Result elements modify their behavior to display such results if and only if the following conditions are all met:

- The Time History Result element is located in the outer (parent) model (i.e., not in a SubModel);
- Within the Result Properties dialog, "SubModel Time" is selected for the **Time Display Setting**.
- A single result is selected for display in the Result element. The selected result must reference a Time History Definition output from a SubModel. The **Output Definition** for the output must be "Realization Histories" (for this to occur, the SubModel must be configured to run multiple realizations);

  Read more: Creating the Output Interface to a SubModel (page 925).

- The SubModel whose result is being referenced is located within the main model (i.e., it cannot be nested inside another SubModel); and
- The outer (parent) model is configured to run multiple realizations.

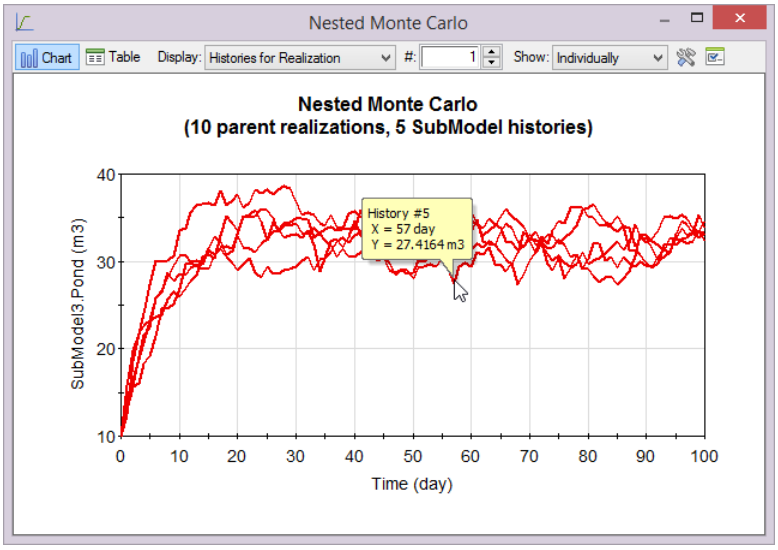
Under these circumstances, special **Display** options are provided at the top of the Time History display dialogs:

- Histories for Realization

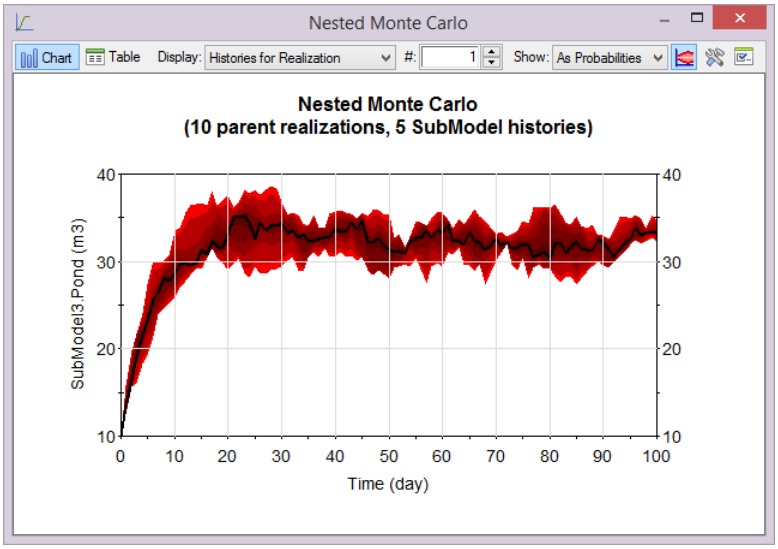
- Statistic for each Realization
- Histories for all Realizations

The various displays (and what they represent) are as follows:

Histories for Realization. "Realization" refers to the realization of the parent model (which you must select). Since each realization of the parent model produces multiple histories for the SubModel, showing "Histories for Realization" simply displays multiple SubModel histories for the selected parent model realization. The chart display looks like this:



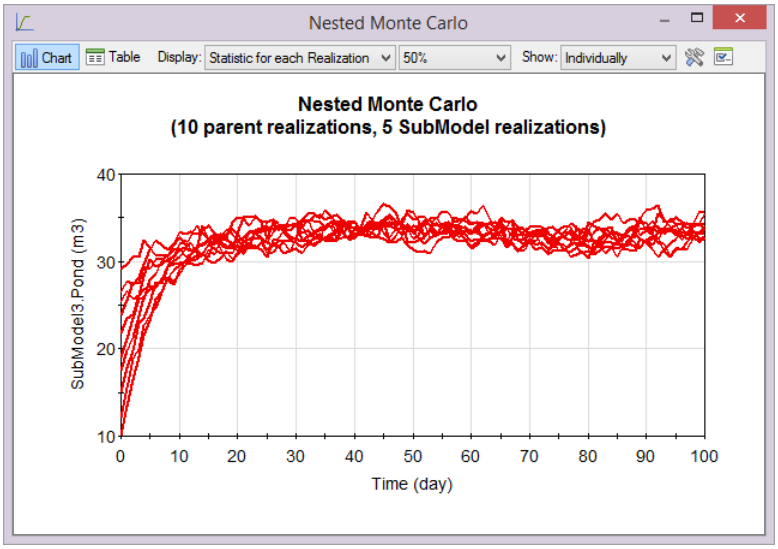
Note that the display also provides a separate field (**Show**). In the figure above, the SubModel histories (in this case, 5) for the specified parent realization (in this case, #1) are being displayed "Individually". A second option allows these 5 histories to be displayed "As Probabilities":



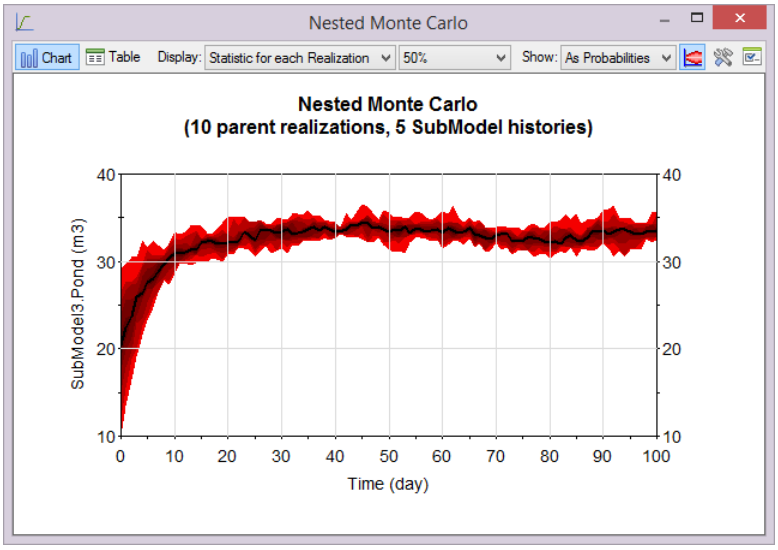
Read more: Viewing Probability Histories for Multiple Realizations (page 555).

Statistic for each Realization. "Realization" refers to the realization of the parent model. Since each realization of the parent model produces a single statistic (which you must specify) for the SubModel histories, showing

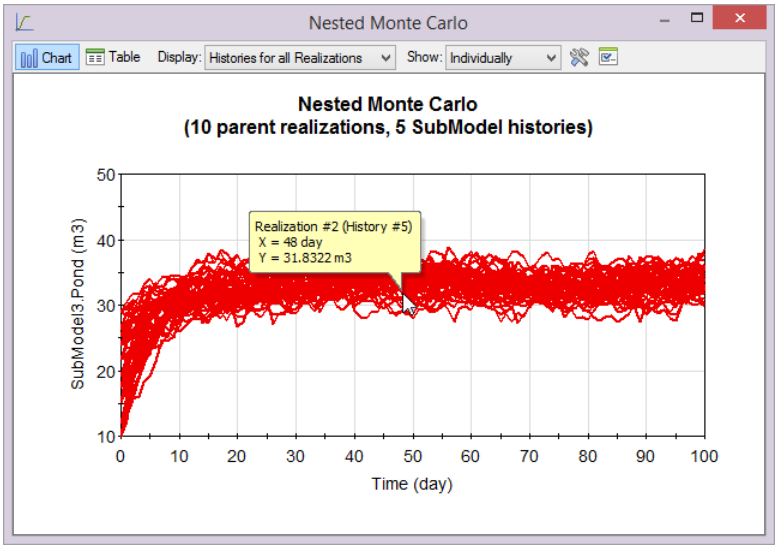
"Statistic for each Realization" displays a single time history for each realization of the parent model. Each individual realization represents a statistic computed using the SubModel histories.The chart display looks like this:



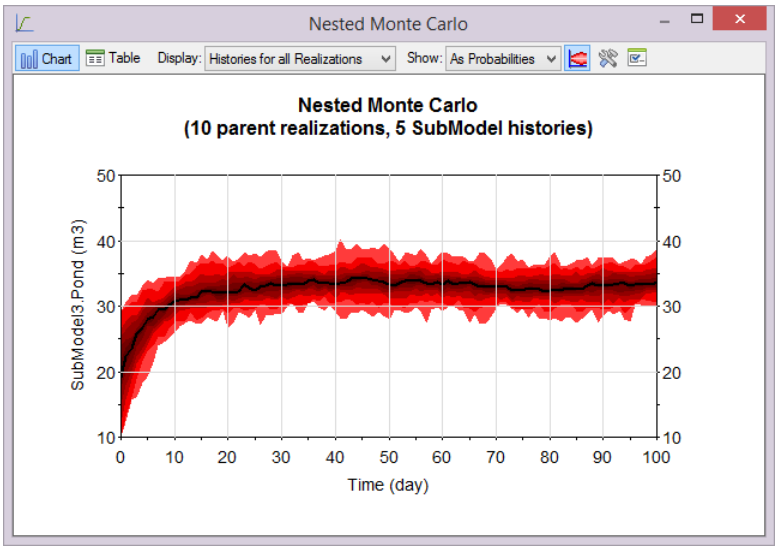
In the figure above, there are 10 realizations shown (one for each parent realization). Each realization represents the 50th percentile computed using the 5 SubModel histories that were produced for that parent realization. In this case, each parent realization is being displayed "Individually". Note that the display also provides a separate field (**Show**). A second option allows these 10 parent realizations to be displayed "As Probabilities":



Histories for all Realizations. Since each realization of the parent model produces multiple histories for the SubModel, showing "Histories for all Realizations" simply displays all SubModel histories for all parent realizations. The chart display looks like this:



In the figure above, there are 50 histories shown (5 SubModel histories for each of 10 realizations of the parent model). In this case, each history is being displayed "Individually". Note that the display also provides a separate field (**Show**). A second option allows these 50 histories to be displayed "As Probabilities":



Note: If you have applied screening to the inner model (the SubModel), the screening is ignored for nested Monte Carlo runs. However, if you have applied screening and/or categories to the parent model, the screening is used to create the displays.

Read more: Classifying and Screening Realizations (page 533).

An example illustrating how nested Monte Carlo simulation is carried out (SubModel_2.gsm) can be found in the General Examples/SubModel folder in your GoldSim directory.

Read more: SubModel Example: Explicit Separation of Variability from Uncertainty (page 952).

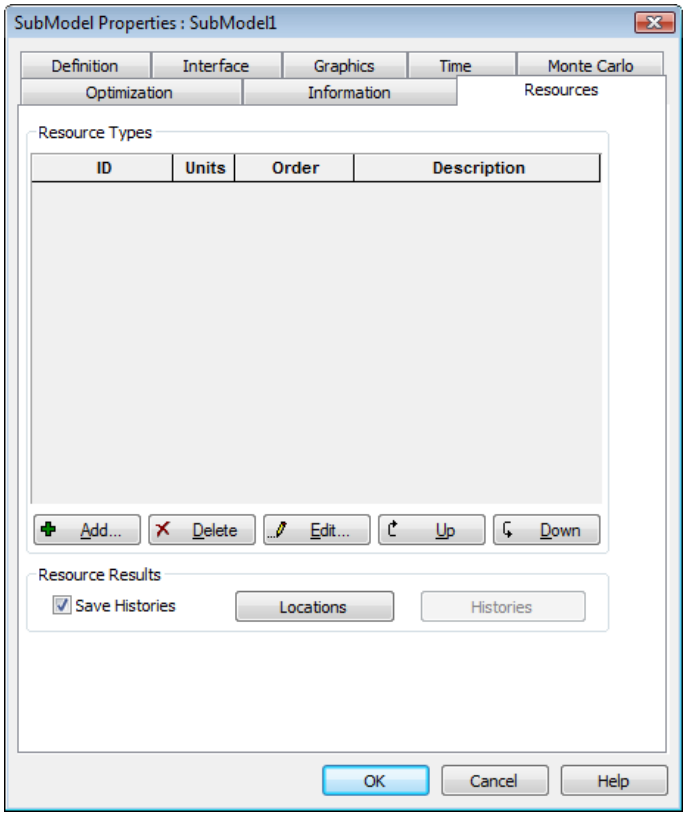
Using Resources Inside a SubModel

Like standalone models and Containers, a SubModel can provide Resources to elements inside it.

Read more: Using Resources (page 781).

To provide Resources to elements inside the SubModel, check the Provide Resources checkbox on the SubModel's **Definition** tab:

When you do so, a Resources tab is added to the SubModel's dialog:



This tab can be used to create Resource Types (and global stores) for use inside the SubModel.

Several points should be noted regarding the use of Resources inside SubModels:

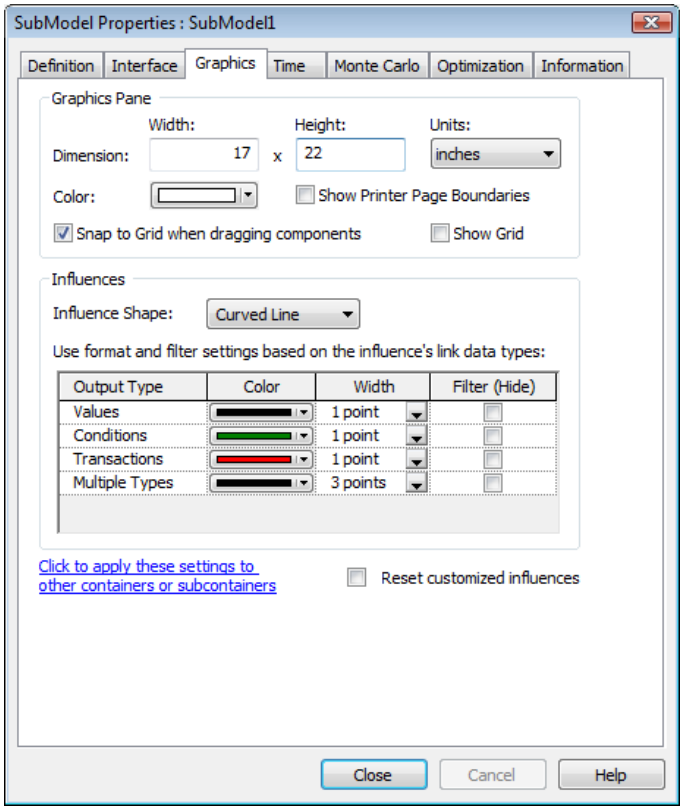
- Elements inside a SubModel have no access to Resource Types or Resource Stores outside of the SubModel. Resources in the SubModel are completely separate and independent from Resources in the parent model.
- If you create a SubModel by importing a GoldSim model, and that model has Resources defined, the **Provide Resources** checkbox will automatically be checked, and any Resource Types and Stores defined in the original model will become part of the SubModel.

  Read more: Importing SubModels (page 948).
- If you export a SubModel, any Resource Types and Stores in the SubModel will become available in the exported standalone model.

  Read more: Exporting SubModels (page 946).

Controlling the Appearance of the Graphics Pane for a SubModel

The **Graphics** tab of the SubModel dialog provides access to options for controlling the appearance of elements inside the SubModel:



The top portion of the dialog (the Graphics Pane section) is used to define the size of the graphics pane, a background color for the graphics pane, and whether or not a grid is displayed.

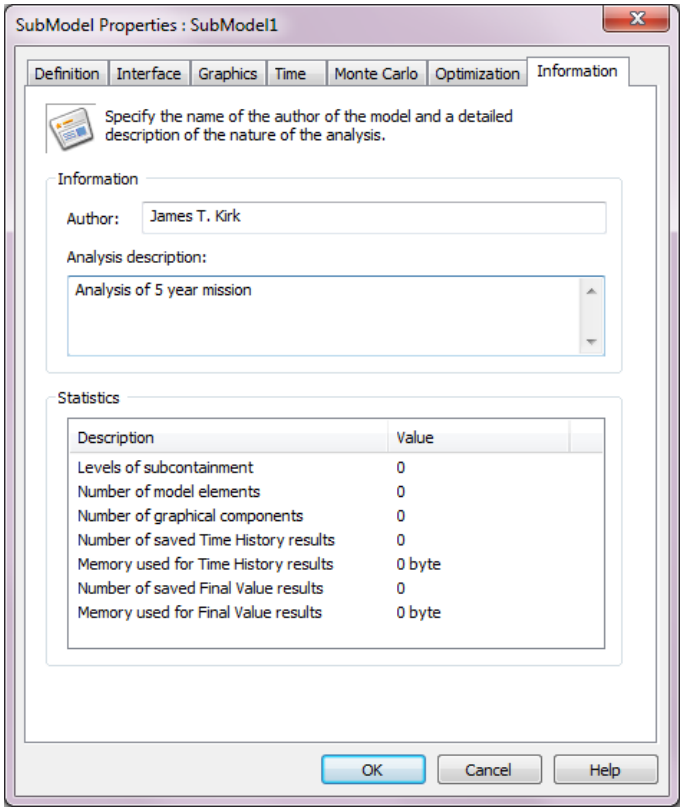
Read more: Adjusting the Size of the Graphics Pane (page 388); The Graphics Pane Grid and Background (page 386).

The bottom portion of the dialog (the Influences section) is used to specify the default shapes for influences in the SubModel, and to define whether and in what manner influences within the SubModel are filtered (hidden). The options in this portion of the dialog are quite important, as they can be used to ensure that your models are easy to view and understand.

Read more: Links and Influences (page 86); Editing the Appearance of Influences (page 389); Filtering Influences (page 392).

Viewing and Editing SubModel Summary Information

The **Information** tab of the SubModel dialog displays some summary information regarding the SubModel:



The top part of the dialog provides two fields (**Author** and **Analysis description**) which allow you to identify the author and provide a brief description for your SubModel.

The bottom part of the dialog provides some useful summary statistics for the SubModel, including the number of model elements and graphical components, and the degree of subcontainment. The number of levels of subcontainment does not refer to the number of Containers; it refers to the number of hierarchical levels of Containers. For example, if the SubModel A contained Containers B and C, A would have 1 level of subcontainment; if the SubModel contained B, and B contained C, it would have 2 levels of subcontainment.

The dialog also indicates the total size of the results being saved for all elements within the SubModel. This information can be very useful in helping you to manage the size of the model file.

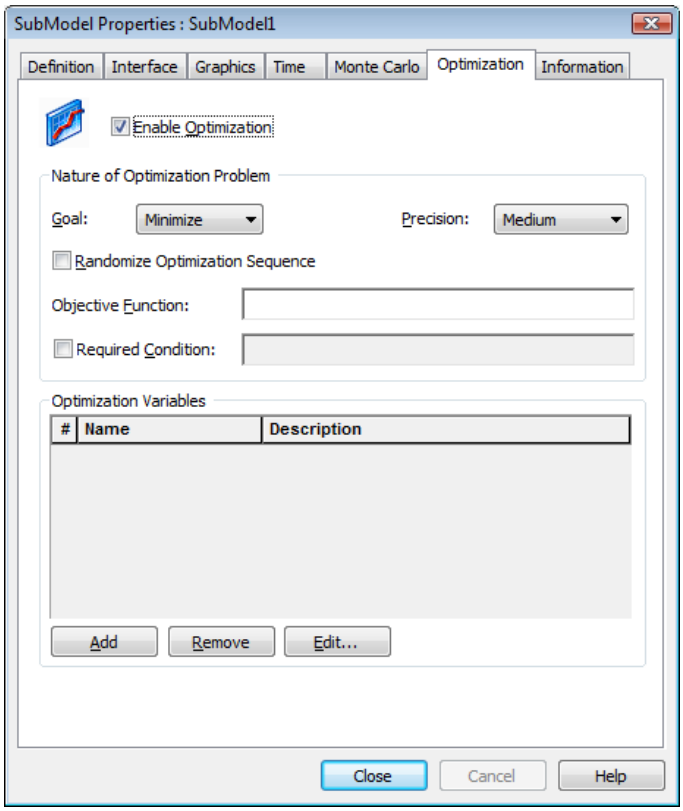
Running an Optimization Within a SubModel

One of the applications for a SubModel is to carry out a dynamic optimization (i.e., at specified times) during a simulation. For example, imagine a situation where you were simulating the operation of a facility over a period of one year. Every month, the operators make a decision based on the current state of the system. This decision is based on a simple optimization analysis using currently available data (i.e., at every simulated month during the simulation). The optimization chooses the optimum values of a few control variables that they will use for the next month.

To represent this in GoldSim, you would need to specify the **Solution Type** for the SubModel as "Optimization", and the SubModel itself would represent the optimization calculations carried out by the simulated operator.

Note: If a SubModel is set to carry out an optimization, it cannot contain any Spreadsheet elements.

When you select "Optimization" for the **Solution Type**, GoldSim enables optimization for the SubModel and immediately switches to the **Optimization** tab for the SubModel:



This is the standard optimization dialog that is also used for stand-alone models, and is described elsewhere in detail.

Read more: Running an Optimization (page 486).

Several points regarding use of this dialog for SubModel optimization, however, are worth noting:

- Optimization Variables cannot be defined in terms of variables that are outside of the SubModel. That is, initial values, upper bounds and lower bounds for Optimization Variables cannot be defined in terms of a variable on the input interface to the SubModel.
- In most cases, all of the Optimization Variables (and perhaps the Objective Function) should be placed on the output interface to the SubModel (since the purpose of the SubModel is typically to determine these variables). When running an Optimization in a SubModel, the only applicable result type is Final Value. At the end of the optimization, the SubModel outputs the final (optimized) values of the Optimization Variables.
- Optimizations can sometimes fail or may be unable to converge. GoldSim provides different types of error messages when carrying out

optimizations. If an optimization of a SubModel fails, a fatal warning message will be displayed (and the simulation is halted). If the optimization completes, but cannot fully converge, a message will be written to the run log (but the simulation will not be halted).

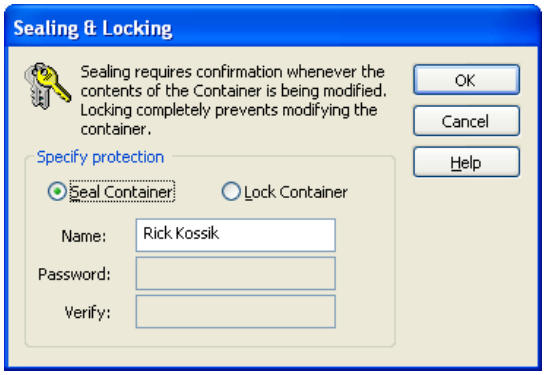
Read more: Understanding Optimization Warning Messages (page 495).

Protecting the Contents of a SubModel

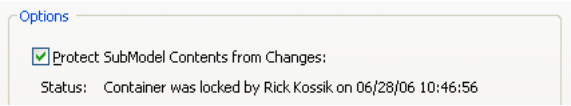
In some cases, you may want to protect the contents of your SubModel by sealing or locking it. SubModels can be sealed and locked in the same manner as Containers.

Read more: Sealing and Locking Containers (page 147).

To access the sealing/locking dialog for a SubModel, check the **Protect SubModel Contents from Changes** checkbox in the SubModel dialog. When you do so, the Sealing/Locking dialog will be displayed:



After you seal or lock the SubModel, the protected status of the SubModel is displayed in the SubModel dialog:

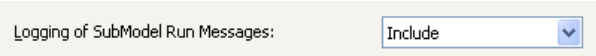


Controlling How Run Messages are Logged for a SubModel

When using SubModels, you can control whether or not Run Log messages related to the SubModel are written to the parent model's Run Log.

Read more: The Run Log (page 506).

This can be controlled from within the SubModel dialog:



By default, **Logging of SubModel Run Messages** is set to "Include". Optionally, however, you can set this to "Ignore".

When SubModel messages are sent to the Run Log, it is clearly indicated which model or SubModel generates the message.

Exporting SubModels

In some cases, you may want to run a SubModel as a standalone model. This is useful, for example, if you want to debug, test, and/or refine it separate from the parent model. Although you can't run the SubModel directly from within a parent model, you can export it, run it separately, and then import it back into the parent model. You may also choose to export a SubModel so that you can subsequently import it into a different model.

Read more: Importing SubModels (page 948).

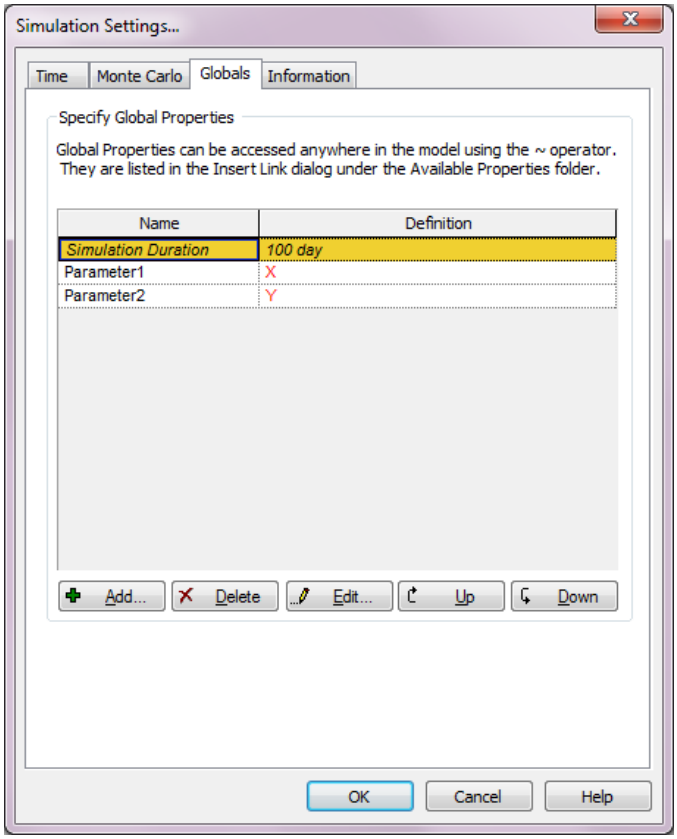
You can export a SubModel by pressing the **Export as .GSM…** button in the SubModel dialog. When you do so, you will be prompted for a file name (and location) for the saved file.

When exporting a SubModel, several points should be noted:

- All of the options (specified in the parent model's Options dialog) are exported with the new model.
- All user-defined units and array label sets specified in the parent model are exported with the new model.
- If you export a SubModel, any Resource Types and Stores in the SubModel will become available in the new model.

  Read more: Using Resources Inside a SubModel (page 942).

In addition, the input and output interfaces are exported to the stand-alone model. The output interface is not accessible in the stand-alone model, but the input interface is accessible, and can be edited. In particular, the input interface is accessible via the **Globals** tab in the stand-alone model's Simulation Settings dialog:



Read more: Defining and Referencing Global Properties (page 443).

In the example shown above, Parameter1 and Parameter2 in the SubModel were referencing outputs X and Y in the parent model. Once the SubModel was exported, of course, it no longer had access to X and Y "outside" of the model (since there is no "outside" to a stand-alone model). Hence if you tried to run the stand-alone model an error would be displayed.

In order to run the stand-alone model, the Definitions for the Globals would need to be changed to values. Note, however, that the original exported

Definitions are saved with the file separately. Therefore, if you subsequently import the stand-alone model into another model (as a SubModel), the original input (and output) interface that was exported will be automatically recreated.

Note: You can only export a SubModel that has values or conditions on the input interface. If the input interface contains Time Series Definitions, Lookup Table Defintions or Distribution Definitions, you will not be able to export the model (as these could not be represented as values on the **Globals** tab.

If the exported SubModel's input interface included externally defined Run Properties (e.g., such as Simulation Duration), these interface items will appear in the Globals list. However, they will not be used by the stand-alone model, and cannot be edited. They can, however, be deleted.

Importing SubModels

In some cases, you may want to import an existing stand-alone model into another file as a SubModel. This is useful, for example, if you want to export a SubModel to test (and refine it) outside of the parent model, and then import it back into the parent model.

Read more: Exporting SubModels (page 946).

To import a stand-alone model into your current model as a SubModel, you must insert a SubModel from the Insert Element menu. When you insert a SubModel, you are presented with a dialog asking if you want to create a new (empty) SubModel, or create the SubModel by importing an existing standalone model. You should choose the latter. When you do so, you will be prompted for a filename and location. After selecting it, the specified file will be inserted into the parent model as a SubModel.

There are several rules that must be met in order for the model to be successfully inserted as a SubModel:

- The stand-alone model must have been saved using the same version of GoldSim that is trying to import it.
- The stand-alone model must be in Edit Mode.

Note: If the model being imported contains Scenarios, the scenario information will be removed when it is imported and only the information pertaining to the Active Scenario will be imported.

Read more: Creating, Running and Comparing Scenarios (page 463).

When importing a model, all of the information on the Globals tab of the Simulation Settings dialog in the stand-alone model is imported into the input interface for the SubModel.

Read more: Defining and Referencing Global Properties (page 443).

Note: If the model being imported was previously exported from a SubModel to a stand-alone model, the original exported Definitions for the input and output interface will be automatically recreated when the model is imported to recreate it as a SubModel. Any changes to existing input Definitions will be ignored (the Definitions that existed when the model was exported will be used). Any new input Definitions that were added after the model was exported, however, will be added to the interface.

If you import a GoldSim model as a SubModel, and that model has Resources defined, the **Provide Resources** checkbox will automatically be checked, and any Resource Types and Stores defined in the original model will become part of the SubModel. The Resources in the SubModel and those in the parent model are completely separate and independent. Elements in the SubModel cannot see or interact with Resources in the parent model.

Read more: Using Resources Inside a SubModel (page 942).

Because the parent model and the imported model potentially could have conflicting definitions (for units, array label sets and version stamps), the following rules are followed during the import to ensure consistency.

Ensuring Consistency of Array Label Sets

When importing an existing model as a SubModel, all of its array label sets are imported into the parent model. Therefore, existing sets and imported sets must be merged. The following rules apply when importing array label sets:

- Any set that exists in the imported model but not in the parent model is automatically added to the parent model.
- If the imported model and the parent model have sets with identical names (set names are not case-sensitive) that are not otherwise identical (i.e., they must have the same number of items with the same index names), an error is displayed and the import of the model will fail.

Read more: Understanding Array Labels (page 727).

Ensuring Consistency of Units

When importing an existing model as a SubModel, all available user-defined units in the existing model are imported into the parent model. Therefore, existing units and imported units must be merged. The following rules apply when importing units:

- If the imported unit name is unique, it is added to the parent's unit list.
- If the imported unit name is identical to an existing unit name, and the units have the same dimensions, but are assigned to different categories, then the unit is assigned to the category specified by the parent model.
- If the imported unit name is identical to an existing unit name, and the units have the same dimensions, but are assigned different conversion factors, then the conversion factor is assigned the value specified by the parent model.
- If the imported unit name is identical to an existing unit name, but the units have different dimensions, an error is displayed and the import of the model will fail.

Read more: Creating New Units (page 402).

Ensuring Consistency of Versioning

When importing a model file that includes versioning information, GoldSim evaluates the version stamps and compares them against the version stamps in the existing model (if it uses versioning).

If all version stamps are identical (i.e., all properties must match: stamp number, name, description, user name), then the imported model elements keep their individual version logging information. Globally logged information from the imported model, however, is lost.

If the version stamps are not identical, or if the existing (parent) model does not use versioning, all version information is removed from the imported model and its elements.

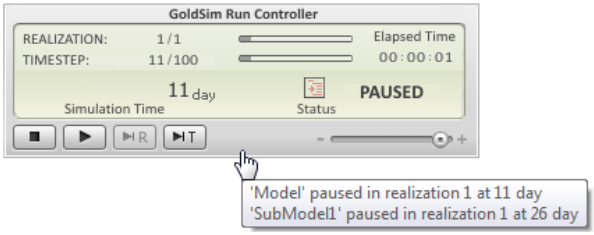
Read more: Tracking Model Changes (page 963).

Interrupting and Pausing a Simulation Within a SubModel

When you pause a model using the Run Controller or via an Interrupt element message, if the model was in the middle of computing an element within a SubModel, GoldSim will pause the model at that point.

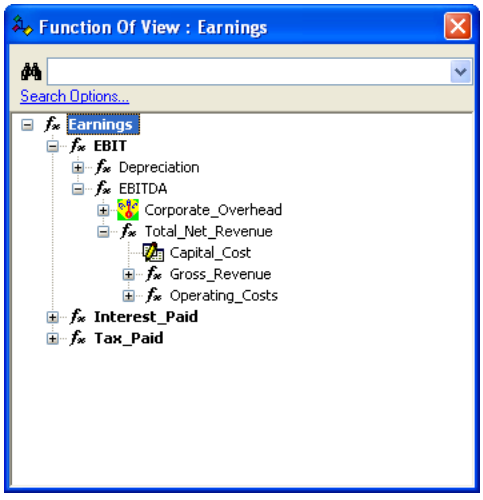
Read more: Pausing and Stepping through a Simulation (page 460); Continuing, Pausing and Aborting a Simulation After an Interrupt (page 375).

When a model is paused, the Run Controller always displays the status of the outermost parent model. However, if a model has been paused in the middle of a SubModel, when you place your cursor over the Run Controller, it will display the status of the SubModel(s) also:



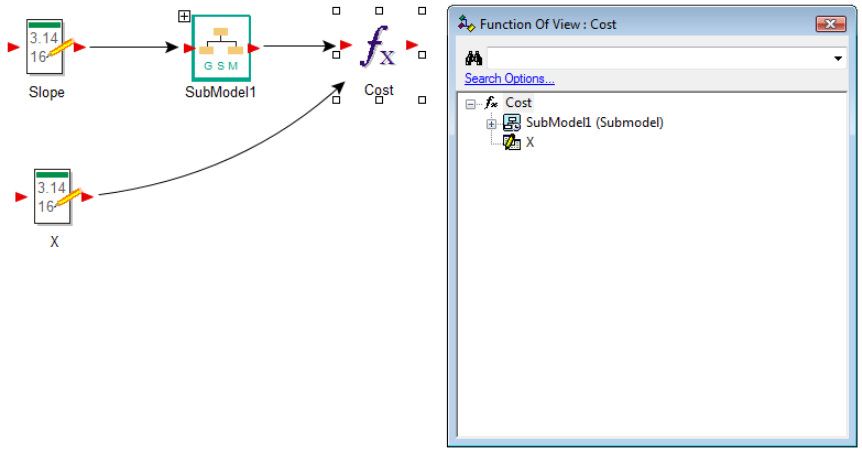
Viewing Element Dependencies Within a SubModel

In complex models, it is often useful to explore the interdependencies of the various elements (i.e., who affects who). GoldSim provides two very powerful utilities for doing this: the ***Function Of View***, and the ***Affects View***. If you right-click on an element in either the graphics pane or the browser (to access the context menu) and select **Function Of…** or **Affects…**, a floating browser window is displayed that shows the element dependencies in the form of a tree:



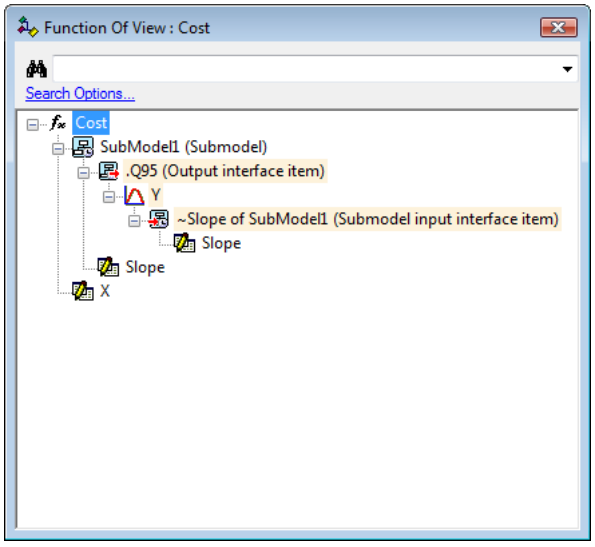
Read more: Viewing Element Dependencies (page 117).

If the tree includes a SubModel, the SubModel is effectively treated as an element (this is in contrast to a Container, which generally would not show up in the tree at all). That is, the SubModel is treated as any other element with inputs and outputs would be treated. With regard to the dependency tree, it is simply a "custom" element. In the example below, the Expression Cost is shown to be a function of X and the SubModel:



In many instances, however, you may be interested in looking inside the SubModel to view the dependencies. That is, it may not be enough to know that Cost is a function of SubModel1; you may want to explore exactly what Cost is a function of within the SubModel.

To facilitate this, GoldSim allows you to expand SubModels within a Function Of or Affects tree:



When GoldSim displays items inside a SubModel in this way, it always does the following:

- The SubModel is clearly labeled as such.
- The contents of the SubModel in the tree are shaded (orange) to indicate that these items are inside the SubModel. (If there are

SubModels within SubModels, different shades are used to indicate this).

- In addition to showing the elements inside the SubModel, GoldSim also displays any items on the input and output interfaces that may be involved (and these are clearly labeled as such). In the example above, the tree is indicating that Cost is a function of the "Q95" output on the SubModel interface, which is a function of the element Y inside the SubModel, which is a function of the "~Slope" input on the SubModel interface, which is a function of the Slope element outside of the SubModel.

SubModel Examples

Several example models illustrating the use of SubModels are described below. These examples can be found in the General Examples/SubModel folder in your GoldSim directory.

These examples cover the four most common applications for SubModels.

Read more: What Can I Do With a SubModel? (page 914).

SubModel Example: Manipulation of Monte Carlo Statistics

A common use of SubModels is to manipulate Monte Carlo statistics. That is, after carrying out a Monte Carlo simulation, you may want to carry out further calculations using the statistical outputs of the simulation. For example, you may want to carry out a calculation that is some function of the mean and the 95th percentile of a particular output in a Monte Carlo simulation.

Without the use of SubModels, the only way to accomplish this is by exporting results from a Monte Carlo simulation manually to another application (e.g., a spreadsheet or a separate GoldSim model). With SubModels, this is easily accomplished within a single GoldSim model by inserting a SubModel (that is a Monte Carlo simulation) into an outer model (that is static and simply manipulates the statistical outputs of the inner model).

Example model SubModel_1.gsm (in the General Examples/SubModel folder in your GoldSim directory) provides a simple illustration of such an application. In this model, the SubModel contains a single Stochastic. The SubModel carries out a Monte Carlo simulation. The outer model is a static deterministic simulation that contains a single Expression element. The Expression is a function of statistics (the mean and 95th percentile) of the Stochastic within the SubModel.

SubModel Example: Explicit Separation of Variability from Uncertainty

Another common use of SubModels is to explicitly separate variability from uncertainty. That is, many models have uncertain parameters as well as variable parameters. An uncertain parameter represents ignorance that can theoretically (but perhaps not practically) be reduced through investigation (e.g., the mean failure time for a batch of light bulbs). Variability is inherent in the system (e.g., the distribution of failure times for a batch of light bulbs) and cannot be reduced.

It is often valuable to explicitly separate uncertainty from variability in a model. With SubModels, this is accomplished by inserting a "variability" SubModel (e.g., a dynamic Monte Carlo simulation) within an outer "uncertainty" model (e.g., a static Monte Carlo simulation). This is referred to as nested Monte Carlo simulation.

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 933).

Example model SubModel_2.gsm (in the General Examples/SubModel folder in your GoldSim directory) provides a simple illustration of such an application. In this model, a nested Monte Carlo simulation is used to calculate the confidence

in the performance of a system of light bulbs. The outer model samples two probability distributions for values that describe the shape of the failure distribution for the light bulb. The fact that these are distributions indicates that we are uncertain about the precise shape of the failure distribution. The inner model then simulates the performance of the light bulbs (whose lifetime is sampled from a failure distribution with the (uncertain) variables provided by the outer model). The result of this type of analysis is a probability of probabilities (e.g., what is the chance that the probability of the light bulbs lasting less than 6000 hours will be less than one?).

SubModel Example: Probabilistic Optimization

Another common use of SubModels is to optimize a probabilistic model. If you wish to optimize a probabilistic (uncertain) system, the objective function to be optimized cannot be a single deterministic output. Rather, it must be a statistic. That is, if X was an output of a probabilistic model (and hence was output as a probability distribution), optimizing X itself would be meaningless. Rather, you would need to optimize a particular statistic (e.g., the mean or 50^{th} percentile) of the output X.

With SubModels, this is accomplished by inserting a SubModel (e.g., a dynamic Monte Carlo simulation) within an outer model (e.g., a static optimization).

Example model SubModel_3.gsm (in the General Examples/SubModel folder in your GoldSim directory) provides a simple illustration of such an application. In this model, the SubModel is a Monte Carlo simulation containing a single Stochastic. The outer model is an optimization. The variable being optimized is a function of another element in the outer model and a statistic (the 95^{th} percentile) of the Stochastic in the SubModel.

SubModel Example: Dynamic Optimization During a Simulation

Another common use of SubModels is to do a dynamic optimization during a simulation. Imagine a situation where you were simulating the operation of a facility over a period of one year. Every month, the operators make a decision based on the current state of the system. This decision is based on a simple optimization analysis using currently available data (i.e., at every month during the simulation). The optimization chooses the optimum values of a few control variables that the operators will use for the next month.

With SubModels, you could simulate this by inserting a SubModel (e.g., a static optimization) within an outer model (e.g., a dynamic simulation).

Example model SubModel_4.gsm (in the General Examples/SubModel folder in your GoldSim directory) provides a simple illustration of such an application. In this model, the outer model is a dynamic deterministic simulation, and the SubModel is a static deterministic optimization. The objective function in the SubModel is a function of a parameter that is passed in from the outer model (and changes every timestep). Hence, every timestep, the SubModel carries out a new optimization and the results are used in the parent model for the next timestep.

Customized Importance Sampling Using User-Defined Realization Weights

For risk analyses, it is often necessary to evaluate the low-probability, high-consequence portions of the probability distribution of the performance of the system, and that can require carrying out a very large number of Monte Carlo realizations. Because the models for such systems are often complex (and hence

need significant computer time to simulate), it can be difficult to use the conventional Monte Carlo for such risk analyses.

To facilitate this type of analysis, GoldSim allows you to utilize an importance sampling algorithm to modify the conventional Monte Carlo approach so that selected portions of input distributions (which could correspond to high-consequence, low-probability outcomes) are sampled with an enhanced frequency. During the analysis of the results that are generated, the biasing effects of the importance sampling are reversed. The result is high-resolution development of "high-consequence, low-probability outcomes", without paying a high computational price.

Stochastic elements provide built-in functionality to force importance sampling on either the low end or high end of a Stochastic element's range (i.e., the tails of distribution). Timed Event and Random Choice elements provide similar functionality.

Read more: Applying Importance Sampling to a Stochastic Element (page 182); Timed Event Elements (page 333); Random Choice Elements (page 341).

In some cases, however, it may be necessary apply importance sampling not just over the tails, but over user-defined regions of the Stochastic element's range. For example, if it was known that the important results of a model were highly sensitive to values of a particular Stochastic that were close to the 40th percentile, you would want to focus sampling in that region.

GoldSim supports this by providing three specialized functions that provide biased sampling (and compute the appropriate weights) for a targeted range within a probability distribution. All three functions require a Target probability and a Width. For example, if you specify a Target of 0.4 and a Width of 0.05, biased importance sampling will be focused in the region between probability level 0.375 and 0.425. The functions provided by GoldSim are as follows:

ImpProb(Old,Target,Width): Takes a sampled probability (Old) and returns a probability biased toward the specified region (the manner in which this is done is discussed below).

ImpWeight(Prob,Target,Width): Takes a biased probability (produced by ImpProb) and returns a weight.

ImpOld(Prob,Target,Width): Takes a biased probability (produced by ImpProb) and returns the unbiased probability. This is the inverse of ImpProb.

By using these functions in conjunction with GoldSim's option for the user to manually specify realization weights, a customized importance sampling scheme can be implemented.

The steps required to implement importance sampling on a single Stochastic element (e.g., named Stoch1) with a Target of 0.4 and a Width of 0.05 are as follows:

1. Create a Stochastic element (e.g., U) defined as a Uniform between 0 and 1. This will generate a random number.
2. Create an Expression element (e.g., Prob) defined as "ImpProb(U,0.4,0.05)". This will generate biased random numbers with which to sample Stoch1.
3. Create an Expression element (e.g., Stoch1Value) with the same dimensions of Stoch1, defined as

"PDF_Value(Stoch1.Distribution,Prob)". This will generate the biased sample of Stoch1.

Read more: Specialized Functions that Operate on Distributions (page 183).

4. Create an Expression element (e.g., Weight) defined as "ImpWeight(Prob,0.4,0.05)". This computes the appropriate weight to use for the realization.

5. Open the Simulation Settings dialog, and go to the **Monte Carlo** tab. In the Probabilistic Simulation section, check the **Specify Realization Weights** box. An Insert link dialog will appear, and you should select the Weight element defined in the previous step. This then instructs GoldSim how to appropriately weight each realization when computing results (e.g., CDFs).

   Read more: Probabilistic Simulation Options (page 439).

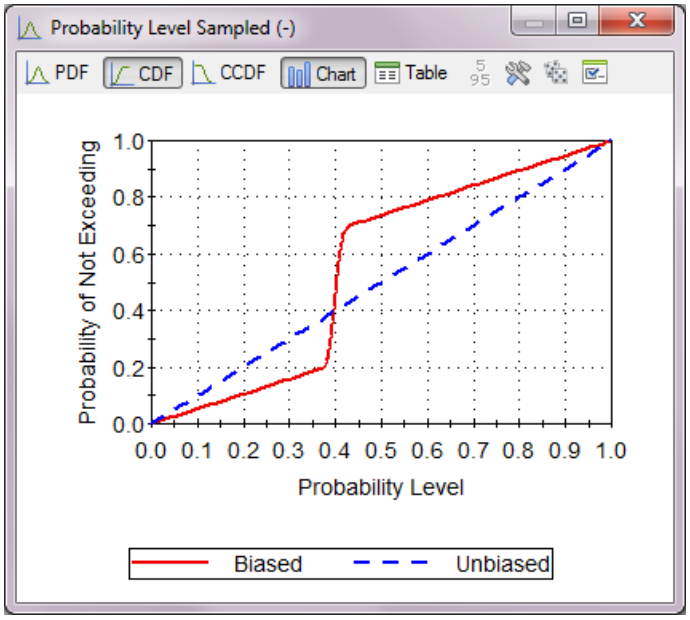
This specific example can be found in the file Custom_Importance.gsm in the General Examples/Stochastic folder of your GoldSim directory.

How does GoldSim actually implement this importance sampling? The implementation is as follows:

During the Monte Carlo simulation half of all the random numbers generated by the importance sampling function (ImpProb) will lie outside of the enhanced-sampling region (defined as the region from Target – Width to Target + Width), and half inside it. The 'outside' and 'inside' regions are sampled differently:

- The half of the samples that are outside the region are distributed uniformly.
- The sampling of the half that are inside of the zone depends on whether the target probability is close to 0 or 1:
  - Normally, these samples have increased sampling rates closer to the target value. The sampling rate within the enhanced sampling zone is distributed according to a truncated normal distribution, with a standard deviation equal to one quarter of the width of the zone.
  - However, if the target probability level is less than the specified width, or greater than (1 – the specified width), then the enhanced samples are also uniformly distributed, though sampled at a higher rate than the samples outside of the region.

The figure below shows how a set of random numbers (generated by sampling from a uniform distribution between 0 and 1) is modified (biased) using the ImpProb function to focus on the specified Target. An unbiased uniform distribution simply produces a CDF that is a straight line with a slope of 1. The biased distribution of random numbers (in this case with a Target of 0.4 and a Width of 0.05 focuses 50% of the samples to fall between 0.375 and 0.425. The remaining 50% falls outside this range (in this case 20% below and 30% above).



Of course, the weights of these samples are adjusted accordingly (using the ImpWeight function) such that in this case, the 50% of the samples falling between 0.375 and 0.425 account for only 10% of the total weight.

The following points should be noted regarding customized importance sampling:

- The Target must be defined as a value between 0 and 1.
- The Width must be between 1E-6 and 0.5.
- Using the approach outlined above, any correlation factor specified for Stoch1 will be ignored (since the value is not generated by the distribution, but via the PDF_Value function).
- This approach can readily be applied to do enhanced sampling of multiple Stochastic elements. It is simply necessary to 1) use a different (uncorrelated) random number (U in the example above) for each element;and 2) multiply all of the independent realization weights together to generate the Realization Weight used in the Simulation Settings dialog. However, when multiple elements use importance sampling it is important not to use narrow sampling regions, as this can lead to numerous 'wasted' realizations that have negligible realization weights.
- The custom importance sampling approach is compatible with the built-in functionality to force importance sampling on either the low end or high end of a Stochastic element's range. Hence, your model could contain some elements that use the "built-in" approach and others that use the custom approach.

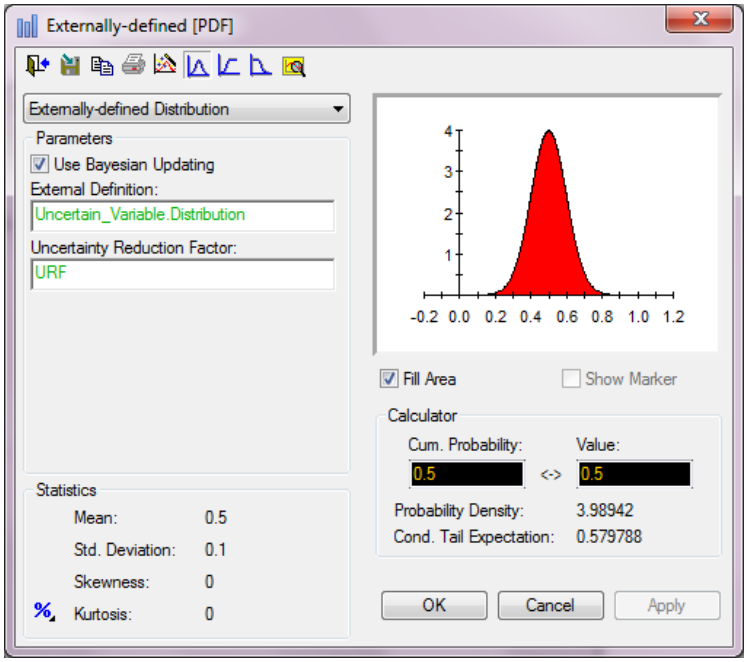
Dynamically Revising Distributions Using Simulated Bayesian Updating

GoldSim provides an advanced feature for dynamically revising distributions during a simulation using "simulated Bayesian updating". This feature is most valuable when using GoldSim to simulate a project of some kind (e.g., developing a new product, building a new facility). It is often the case that as a

project progresses, additional information about uncertain variables is acquired. From the project operator's point of view, their uncertainty is decreasing, and hence as time progresses, the probability distributions used in their model of the project are converging towards the true values of the variables. As their knowledge increases they are better able to make appropriate decisions for the project's path forward.

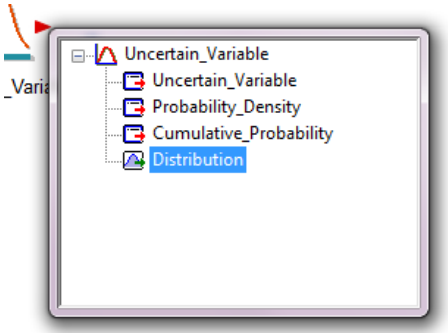
When simulating a project in this manner using GoldSim, the model that is constructed will use Stochastic elements to specify the initial probability distributions of uncertain parameters, and a Monte Carlo sampling process will be used to pick the 'true' values of those parameters. Given that starting point, simulated Bayesian updating can be used to simulate the additional information that will accrue as the project progresses. This in turn can be used to simulate the decisions that would be made based on the new information.

To support this, GoldSim allows you to define an "uncertainty reduction factor" for an uncertain variable. To do so, you must create a new Stochastic and specify its distribution type as an "Externally-defined Distribution":



Read more: Externally-Defined Distribution (page 168).

The uncertain variable (whose uncertainty is being reduced as the model progresses) is entered in the **External Definition** field for this new Stochastic. This must be a link to the Distribution output of a Stochastic element. The Distribution output is a complex output that contains all the information regarding the distribution. It is the fourth output of a Stochastic:



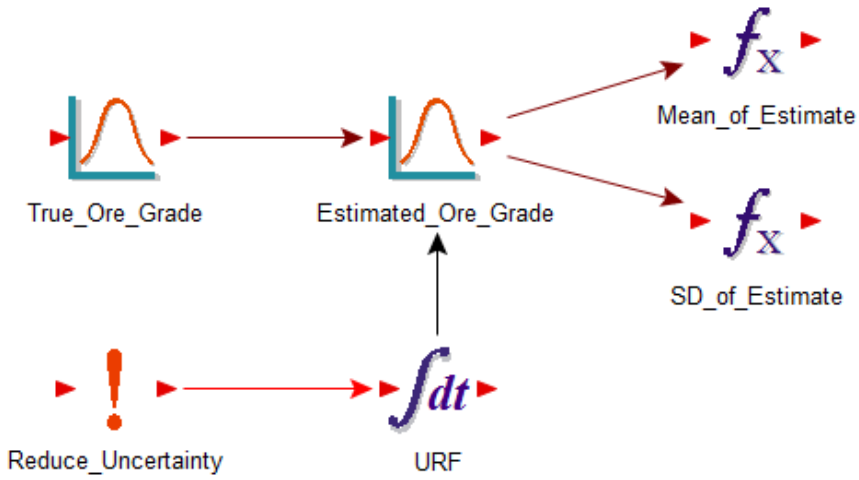
Read more: Stochastic Elements (page 156).

You must then check the **Use Bayesian Updating** checkbox, and then enter a value for the **Uncertainty Reduction Factor**. This dimensionless factor, which must be between 0 and 1, approximates the extent to which the original distribution's standard deviation has been reduced at any point in time. Essentially, this allows you to define how the initial uncertainty about the original distribution is reduced. As such, the **Uncertainty Reduction Factor** will almost always have an initial value of 1 (no reduction of uncertainty), and become smaller as a function of time (typically by changing in discrete steps representing points in time when new information has been obtained).

To understand the mechanics of how simulated Bayesian updating works, let us first examine a simple model that illustrates how the Externally-defined distribution is modified by a time-variable Uncertainty Reduction Factor. Subsequently, we will describe a (simplistic) real-world application.

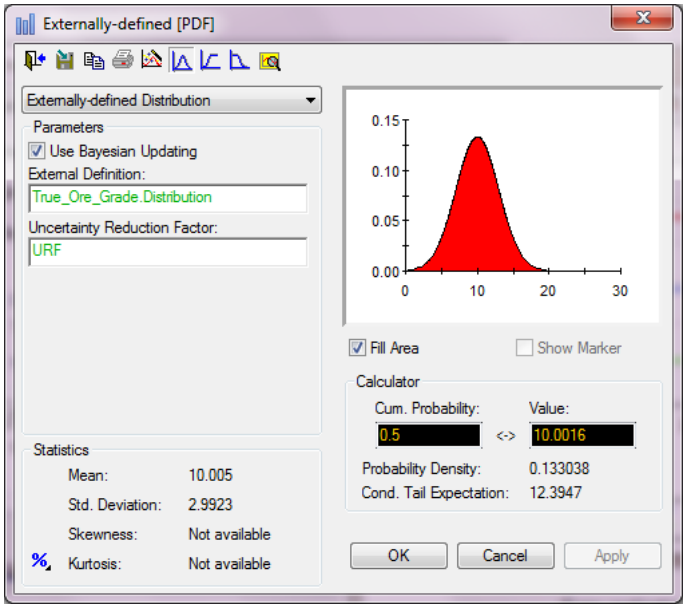
The example model described here can be found in the file URF.gsm in the General Examples/Stochastic folder of your GoldSim directory. It is recommended that your open that file and follow along during the discussion below.

This very simple model looks like this:

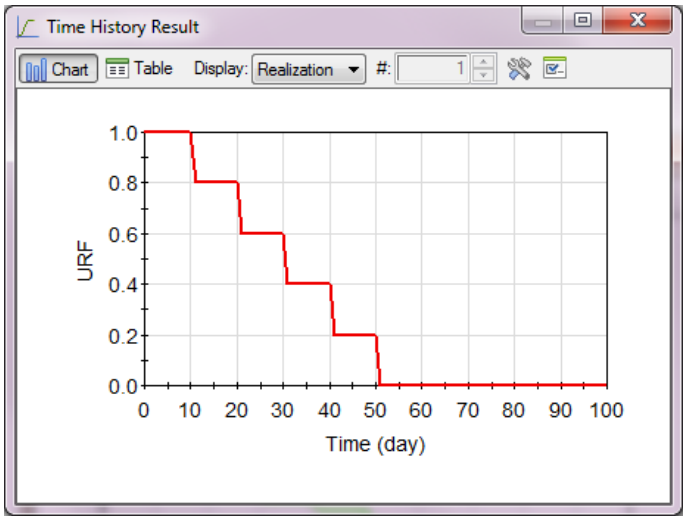


The True_Ore_Grade represents the variable that is being studied (and whose uncertainty is being reduced during the simulated project). In this example, it is a Normal distribution with a mean of 10 and a standard deviation of 3. The Estimated_Ore_Grade represents the updated distribution resulting from studies carried out on the ore grade during the project. It is an External Distribution

whose primary input is the True_Ore_Grade, and whose Uncertainty Reduction Factor is defined by the variable URF:



URF is simply an Integrator with an initial value of 1 (no uncertainty reduction) that is updated by a Discrete Change at discrete points in time (representing studies that have reduced the uncertainty in the ore grade). In this example, URF is simply reduced by a factor of 0.2 every 10 days for a period of 50 days:

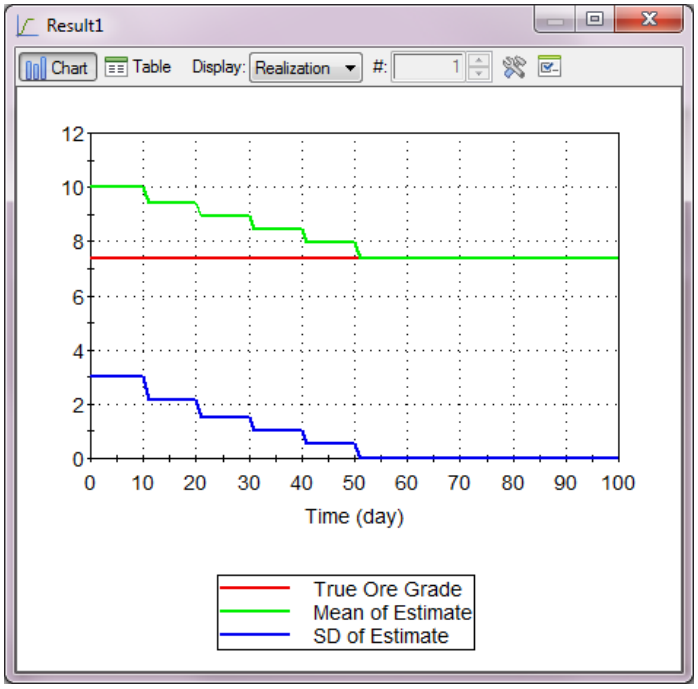


Hence, after 5 days, the uncertainty in the ore grade has actually been reduced to zero (i.e., it is assumed that the ore grade is perfectly known at 50 days).

Given this information, it is instructive to view how the Estimated_Ore_Grade distribution changes as a function of time. Although we can't easily view the distribution in the middle of a simulation, we can examine its statistics and see how they change with time. To do so, we take advantage of two of GoldSim's builit-in functions: PDF_Mean and PDF_SD, which return the mean and standard deviation, respectively, of a specified distribution.

Read more: Specialized Functions that Operate on Distributions (page 183).

The plot below shows how the mean and the standard deviation of the Estimated_Ore_Grade distribution change with time. The plot also includes the True_Ore_Grade:



Every realization of the model produces a different sampled True_Ore_Grade. In this particular realization, the True_Ore_Grade was sampled to be just less than 8. As can be seen, the initial mean and standard deviation of the Estimated_Ore_Grade are 10 and 3, respectively (the same as the statistics for the True_Ore_Grade distribution). However, starting at 10 days, more information is learned about the ore grade. This has two effects on the Estimated_Ore_Grade distribution: 1) the standard deviation is reduced; and 2) the mean approaches the true value. Whenever new information is obtained (in this example, every 10 days), the standard deviation is further reduced, and the mean moves closer to the true value. At 50 days (when it is assumed that "perfect information" is obtained), the standard deviation goes to zero, and the mean converges to the true value.

Now that we have explained conceptually how simulated Bayesian updating is carried out, let's now expand on this example slightly to illustrate how it might actually be used in a real model.

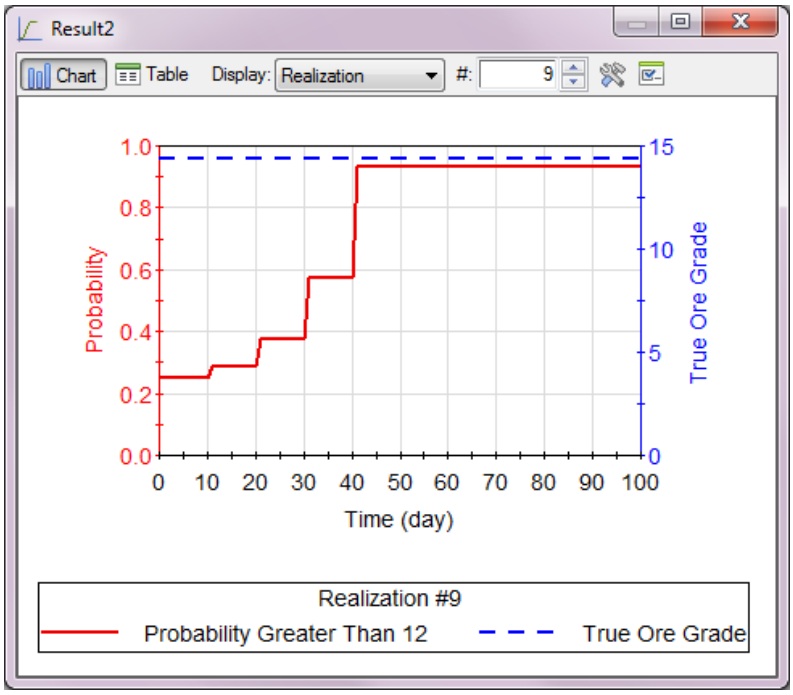
Let's consider the case of a proposed new mine. Initially the mean ore grade may have considerable uncertainty, but as successive studies are carried out, additional information is obtained and the uncertainty about the mean ore grade is reduced. Let's further assume that once you have a particular level of certainty regarding the value, you will act in one way or another.

We will represent this by starting with the previous example, and making one change and one addition:

- We will assume that only four studies are carried out (at 10, 20, 30 and 40 days), each reducing the Uncertainty Reduction Factor by 0.2, such that (as would be realistic) perfect information regarding the ore grade is never obtained.

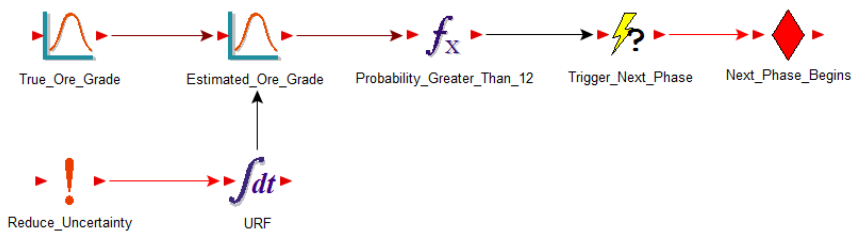
- We will assume that the estimated ore grade must be greater than 12 for the project to proceed. Because the ore grade cannot be known with certainty, however, this decision must be based on probabilities. In particular, when simulating the project, it is assumed that the project will proceed if and only if we have at least 50% confidence that the ore grade is greater than 12 (i.e., we will proceed if the probability of the ore grade being at least 12 exceeds 50%).

The probability of the ore grade being at least 12 is another statistic (that is changing with time as more information is obtained) that can be computed using one of GoldSim's built-in functions (PDF_CumProb). The plot below shows how the probability of the ore grade being at least 12 changes with time for a particular realization:



In this realization, the true value is sampled to be approximately 14.3. Given the original distribution (Normal distribution with mean 10 and standard deviation 3), the original probability of the value being greater than 12 is about 25%. As the uncertainty in the ore grade is reduced, the probability of the ore grade being greater than 12 increases, until at 30 days, it jumps above 50% to 57% (and at 40 days jumps to 93%), and hence the project can proceed.

In this simplified model, when this occurs, an event is triggered which in a real model could trigger the next phase to begin:



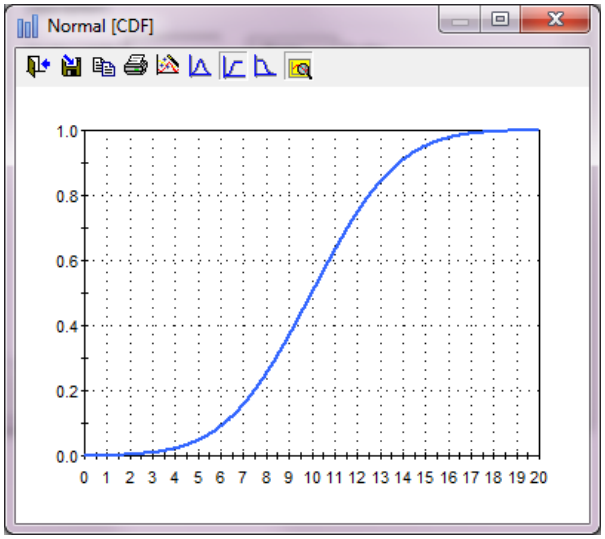
Although this example is simple, it illustrates how simulated Bayesian updating can be used to simulate additional information that is obtained as a project

progresses, and how this can in turn be used to simulate the decisions that would be made based on the new information.

Mathematics of Simulated Bayesian Updating

The conventional approach to sampling an uncertain variable is to first sample a random value (from a uniform distribution between 0 and 1), and then use this value as input to the inverse cumulative distribution function for the uncertain variable.

For example, the CDF for a Normal distribution (with mean 10 and standard deviation 3) looks like this:



To sample this distribution, GoldSim would first sample a random value from a uniform distribution between 0 and 1 (such that all values were equally likely), and use this as the cumulative probability level for the sampled variable. It would then map that probability to a value. For example, if 0.4 was sampled from the uniform distribution, tha would map to a value for the variable of approximately 9 in this case.

To understand how simulated Bayesian updating is carried out, recall that two separate Stochastic elements must be defined to carry it out: 1) a Stochastic representing the "true" value; and 2) a Stochastic (defined as an Externally-defined distribution with a specified Uncertainty Reduction Factor) representing the "updated" distribution. When doing Bayesian updating, the key point is that the random value on the probability axis for the "updated" distribution is not sampled from a uniform between 0 and 1; rather, it is sampled from a beta distribution, whose shape depends on the sampled "true" value and the specified Uncertainty Reduction Factor for the "updated" distribution.

The specific way this is done is as follows:

1. A random number is selected in order to sample the distribution representing the "true" value.
2. In order to sample the "updated" distribution, a beta distribution is created on the probability axis for the distribution. This beta distribution has a minimum value 0 and maximum value 1. Its standard deviation is calculated as the product of the default standard deviation value for sampling the probability axis (0.288675, for a uniform distribution) and the Uncertainty Reduction Factor. The mean of the beta moves linearly from the original mean (0.5 for a uniform distribution) towards the sampled probability level for the original Stochastic as the Uncertainty Reduction Factor ranges from 1 to 0.

Hence, if we assume that the sampled random number for the "true" value is 0.3, and the Uncertainty Reduction Factor is 0.5, then the beta distribution that is created would have a mean of 0.4 (half way between 0.5 and the sampled value 0.3) and a standard deviation of 0.288675*0.5.

3. This beta distribution is then sampled (returning a value between 0 and 1).
4. This value is then used as input to the inverse cumulative distribution function for the original Stochastic.

Note that as the Uncertainty Reduction Factor approaches 0, the "updated" value approaches the "true" value.

The approach described above is used for all continuous distributions. As described above, it effectively increases the likelihoods of "updated" values closer to the "true" result, and reduces the likelihoods of values that are further away from it. This is usually an appropriate approach, as most tests of continuous variables will have a limited accuracy and resolution, and as such, the probabilities of nearby values are correlated. However, for a discrete distribution (e.g. 'is it an apple, an orange, or a pear?') there is likely no correlation between the probabilities of nearby values (and for Boolean, the concept of "nearby" values does not even apply). Accordingly, a different approach (described below) is used for Boolean and Discrete distribution types. However, the approach described above for continuous distributions is also used for the results of binomial, Poisson, and other discrete-valued distributions, as there typically would be a correlation between nearby values for these distributions.

For the Boolean and the Discrete distributions the uncertainty reduction works differently. In these cases, for the "updated" distribution, the probability of the "true" value is increased linearly from its original probability level to a value of 1 as the Uncertainty Reduction Factor drops from 1 to 0. For all other outcomes, the original probabilities are simply multiplied by the Uncertainty Reduction Factor, dropping to 0 as the Factor approaches 0. As was the case for the continuous distributions, sampling is accomplished via an auxiliary probability distribution that modifies the sampled probability levels, but instead of a beta distribution, a cumulative distribution is used in order to have the desired effect.

Tracking Model Changes

GoldSim provides the ability to track changes that you have made to your model file. This feature (referred to as ***versioning***) allows you to quickly determine the differences between the current version of your model file and some previous version of the file.

Providing this configuration management capability is particularly useful for:

- coordinating model changes when multiple people can access and modify the model file;
- as a Quality Assurance/Quality Control feature enabling you to demonstrate and document when and what changes have been made to a model file.

Versioning Overview

Changes to a model file are tracked by creating model file versions. A version is an internal "snapshot" of your model file at a particular point in time. You must tell GoldSim when to take a "snapshot" by creating a version, and assigning it a title (e.g., "1-12-2002", "Initial Model", "Revision A"). Once you have created

at least one version, you can then compare the current model (the model as it exists right now) to any previous model version you have created.

GoldSim can report the differences between the current model and any previous version. Note that GoldSim does not actually tell you *how* a model has changed; in general, it only reports *what* has changed.

For example, if you change some of the inputs to a Reservoir element between versions, GoldSim will report that the element has been changed, but it does not store or report which inputs changed or what their previous values were. Similarly, if you delete an element from a Container, GoldSim will report that an element with a particular ID was deleted from the Container, but it does not record the type of element (or the manner in which the element was defined). This keeps the size of the model file (where all of the changes are stored) manageable.

Generally, you will want to create a version whenever some project milestone has been met. The milestone might be the end of a particular phase of the project, or after you have made some major modification to the model. In some cases, you may want to create versions on a regular basis (e.g., every Friday).

Immediately after you create a version, it may be useful to archive a copy of the model file. If you do so, you will then be able to determine the details of the changes that were made between versions.

Read more: Saving, Opening and Closing GoldSim Files (page 72).

For example, when GoldSim reports that element A was modified between version X and the current version, you can open and compare the file that was archived when version X was created to determine how the element was defined at that point. When you archive a copy of your current model, GoldSim records the name of the archived file (along with the version number) in your current model.

Read more: Changes Tracked Between Versions (page 966).

Enabling Versioning

In order to enable versioning in a file, select **Model|Versioning…** from the main menu. The first time you do this, a message is displayed asking you to confirm that you want to enable versioning.

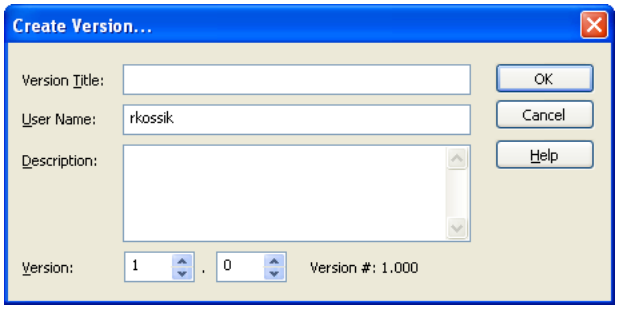
After confirming your request, you will immediately be prompted to create a version of the file (take a "snapshot" of the current model file and "stamp" it as a version).

Once you create the first version, versioning will be enabled, and will remain enabled until you choose to disable it.

Read more: Disabling Versioning (page 966).

Creating Versions

When you first enable versioning, or whenever you choose to create additional versions via the Version Manager , a dialog for creating versions is displayed.



Read more: The Version Manager (page 965).

You must assign a **Version Title** to each version. This Title is not required to be unique (although in most cases it should be). It is used in messages and reports when comparing versions, and should be something meaningful and explicit (e.g., "Draft #1", "Final Model", "Jim's Modifications", "January 12").

The **User Name** will also appear in difference reports and messages, and is intended to identify the model user who created the version. By default, the Windows user name will appear, although you can edit this.

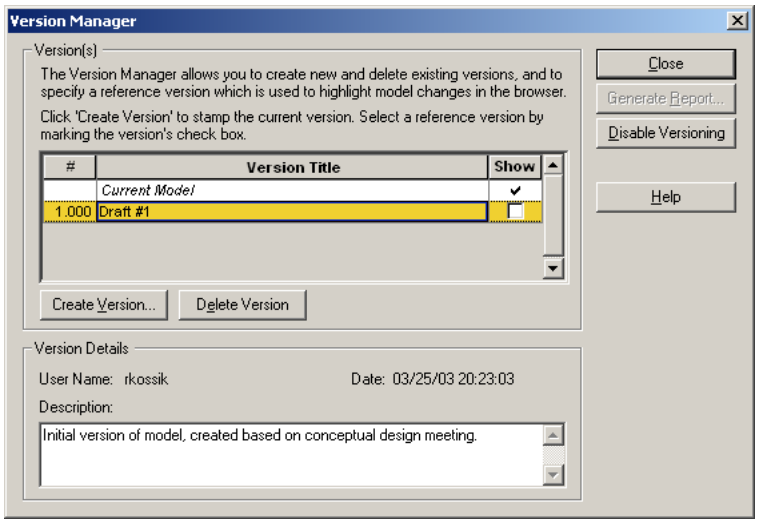
The **Description** field can be used to add descriptive text for each new version.

Finally, you must assign a **Version** number of the form X.yyy (e.g., 1.003). GoldSim ensures that new version numbers are always greater than the previous version number.

The Version Manager

Once you have defined your first version, or if you select **Model|Versioning…** after versioning has already been enabled, the Version Manager dialog is displayed.

The Version Manager lists all the versions that you have created. If you select one of these versions, the details of that version (user name, date created, and description) are displayed at the bottom of the dialog:



You can edit the **Description** here, but cannot change any other details about the version.

You can create new versions by pressing the **Create Version…** button, which provides access to the Create Version dialog.

Read more: Creating Versions (page 964).

Deleting a Version

You can delete an existing version by selecting it in the Version Manager and pressing the **Delete Version** button. Note, however, that when you delete a version, you also automatically delete all previous versions.

When you delete a version, the version number is never used again. For example, if you had a single version (1.000), and you deleted this version and then re-enabled versioning and created another major version, it would be numbered as 2.000.

If you choose to delete the most recent version (and hence all versions in the model), GoldSim will automatically disable versioning (until you enable it again by creating a new version).

Specifying the Reference Version

Once you have created at least one version, you can then compare the current model (the model as it exists right now) to any previous model version you have created. In order to do so, you must identify the ***Reference Version*** (the version to which the current model is compared).

This is done by selecting the checkbox to the right of the version in the Version Manager. Only one checkbox can be selected at a time (i.e., there can be only one Reference Version).

You can readily change the Reference Version while you are viewing differences.

Read more: Displaying Version Differences (page 967).

Disabling Versioning

You disable (remove) versioning by pressing the **Disable Versioning** button in the Version Manager dialog. When you disable versioning, all versioning information is deleted (and cannot be recovered). Hence, before doing so, GoldSim will ask you to confirm that you want to delete all versioning information.

Changes Tracked Between Versions

When versioning is enabled, GoldSim can report the differences between the current model and any previous version. GoldSim does not actually tell you *how* a model has changed; it only reports *what* has changed. For example, if you modify an input field of an element, GoldSim does not store what field was edited or what the previous value was. Rather, it simply reports that "one or more of the element's attributes have been changed".

The changes that are logged by GoldSim can be divided into two categories: global changes and element-specific changes. Global changes pertain to the entire model. Element-specific changes pertain to a particular element.

The following global changes are recorded and stored within the model file when versioning is enabled:

- A new version is created.
- A version is deleted.
- Versioning is disabled.
- The model filename has changed via a Saved As command (the previous filename is recorded).
- The model filename has been changed outside of GoldSim since the file was last opened (the previous filename is recorded).
- The model file has been archived (the archived filename is recorded).
- One or more of the Simulation Settings have changed.
- A user-defined unit has been created or modified.
- An array label set has been added, deleted or changed.
- A global database download has been carried out.
- A file has been locked onto or unlocked by a Spreadsheet, External or File element.
- One or more options in the Options dialog associated with an extension module have changed.

The following element-specific changes are recorded and stored within the model file when versioning is enabled:

- An element is added, moved, deleted.

- An element is cloned or a cloned element is freed.
- The element's ID has been changed (the previous ID is recorded).
- The element's description has changed (the previous description is recorded).
- Any of the element's attributes or input definitions have changed.
- A database download to the element is carried out.
- Data is imported into a Lookup Table or a Time Series element.
- The checkbox in the Options dialog controlling whether or not Timed Events are mapped to the next timestep has changed.

The following points regarding the changes tracked by GoldSim should be noted:

- A Container is considered to be changed if an element is added to, removed from, or deleted from it.
- Locking/unlocking and sealing a Container are not recorded as changes.
- Purely graphical or cosmetic modifications to the model such as changing an element's symbol or adding a graphic or text to the graphics pane are not recorded as changes.
- Adding, deleting or modifying Classification categories are not recorded as changes.
- Adding, deleting or editing a Note to an element is not recorded as a change.
- If an element is changed multiple times between versions, GoldSim records that the element has changed, but does not record the number of times or the number of changes that have been made.
- If an element is changed and then changed back (e.g., if the ID is changed and then changed back to its original ID), GoldSim still records this as a change.

Displaying Version Differences

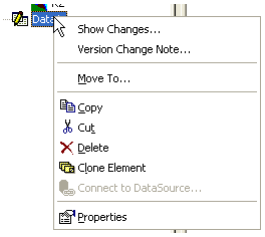
In order to display the differences between the current model and any previous version, you must specify the Reference Version in the Version Manager dialog.

Once a Reference Version is specified, GoldSim identifies changed elements in all browsers as follows:

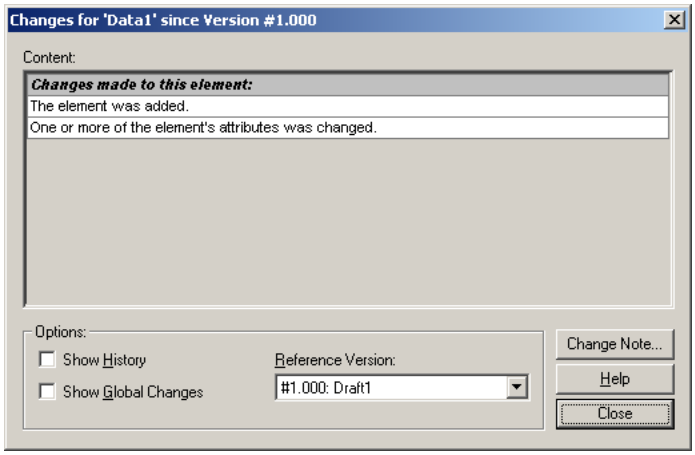
- If an element has been changed since the reference version, the element will appear red in all browsers. (If it is sealed or locked *and* it has changed, it will appear light red.)
- If any element inside a Container has been modified, but the Container itself has not been modified (e.g., no elements have been added or removed from it), the Container will appear blue in all browsers. (If the Container is sealed or locked and elements within it have been changed, it will appear light blue.)

Warning: If a Reference Version is not selected, none of the changed elements will be indicated in the browser. Hence, you must specify a Reference Version in order to view changes.

If you right-click on a colored element (e.g., a red element or a blue Container) in the browser, a menu item, **Show Changes…**, is available toward the top of the menu:

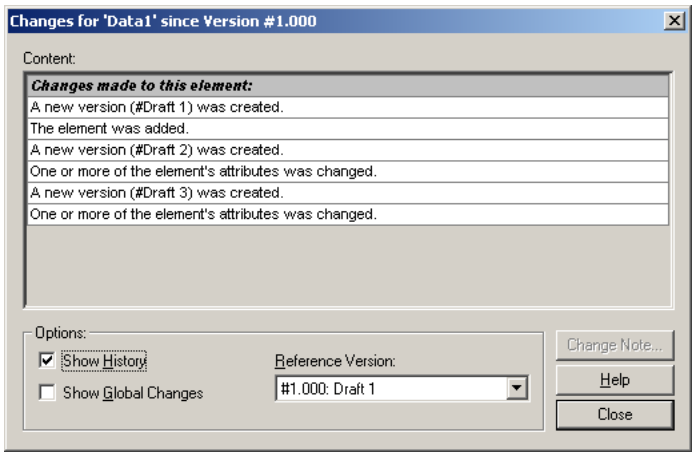


If you select this menu option, the Changes dialog is displayed:



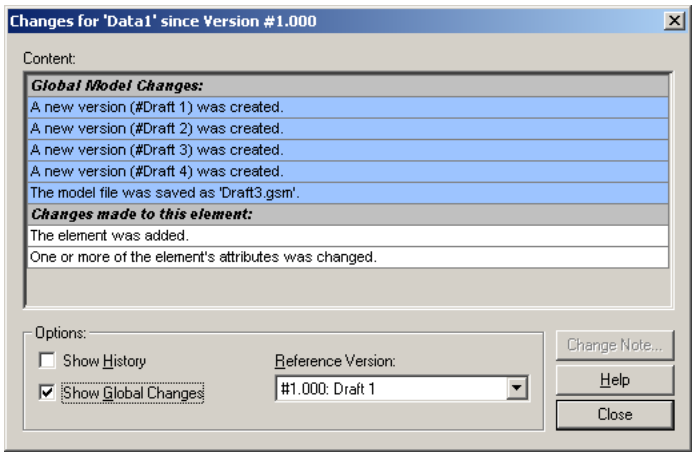
This dialog displays all of the changes that have been made to the element since the specified Reference Version. The Reference Version (selected in the Version Manager) is displayed at the bottom of the dialog . You can change the Reference Version directly from this dialog by selecting a version from the drop-list.

Clicking the **Show History** checkbox shows a history of all the changes between the Reference Version and the current model.

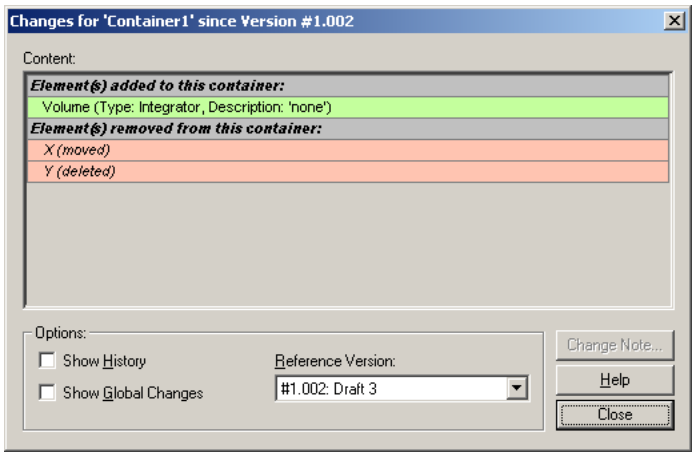


If the **Show History** box is cleared, only a summary of the differences between the Reference Version and the current model are shown. Hence, if the element was changed several different times, the Show History display will indicate this, while the summary display (with **Show History** cleared) will only indicate that the element has changed since the Reference Version.

Clicking the **Show Global Changes** checkbox displays the Global changes to the model (in addition to the changes to elements):



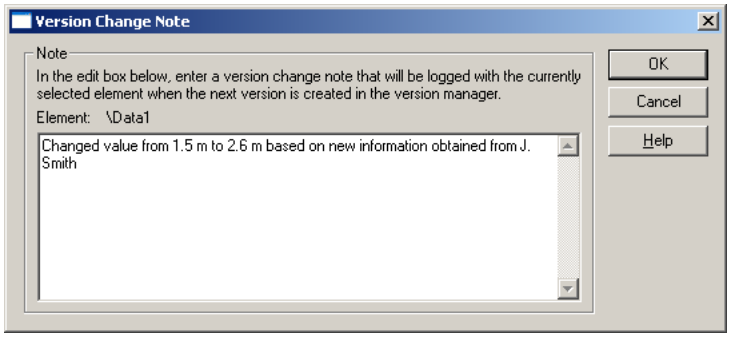
If the selected element is a Container, the Changes dialog identifies any elements added to the Container, any elements removed (or deleted) from the Container, and whether any Container properties (e.g., conditionality settings) have been changed.



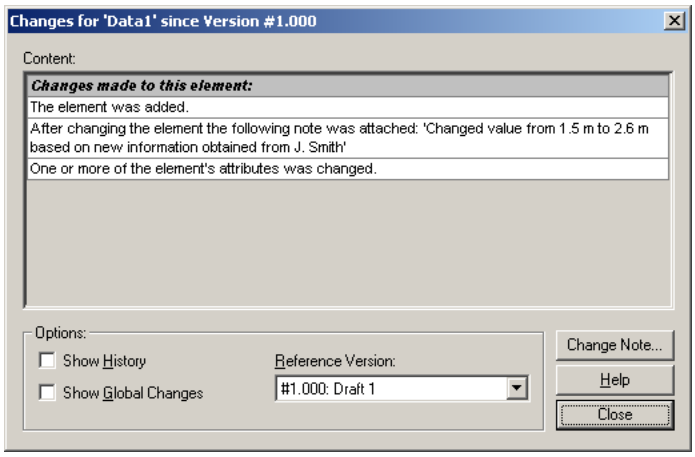
Manually Documenting Changes to an Element

After an element has been changed, you can add a Change Note to the element that will also be displayed in the Changes dialog.

If an element has been changed since the last (most recent) version stamp, the **Version Change Note…** option is available in the context menu for the element, and the **Change Note…** button is available on the Changes dialog for the element. Selecting either of these buttons will display a dialog for adding a change note to the element:



If you add a note , the note is saved with the next version that is created, and is displayed in the element's Changes dialog.

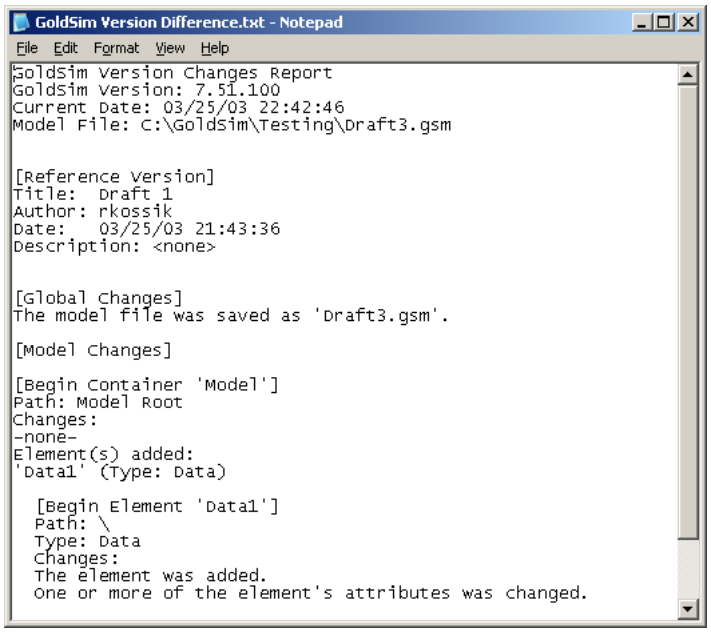


Note: A Change Note is intended to be applied for a particular version. Therefore, whenever a new version is created, the element Change Notes for that version are recorded with the version and then immediately cleared. As a result, if you create a subsequent version, the previous note does not appear in the Changes dialog for the new version.

Generating a Version Report

In some cases, rather than viewing changes to individual elements, you may want to create a report of all of the changes made between the Reference Version and the current model. You can do so by pressing the **Generate Report…** button in the Version Manager dialog.

An ASCII text file summarizing all the changes will be generated and immediately opened in the default application associated with .txt files (e.g., Notepad):



This file is written to the directory containing the model file. If it cannot be saved there (due to access issues), GoldSim will save it to the user's temporary folder (and provide the location of the file in a message).

By default, the name of the Version Report file is "GoldSim Version Difference.txt".

Note: On rare occasions, you may want to instruct GoldSim to insert the model filename into the name of the Version Report filename. To do this, you must edit the Windows Registry. In particular, add a DWORD registry key under HKEY_CURRENT_USER\Software\GTG\Settings named *VersionReportEmbedModelName* and set it to a non-zero value. If you do so, the name of the Version Report file will be "ModelFilename _VersionReport.txt. For example, if the model filename was called "Example.gsm", the Version Report file would be named "Example_VersionReport.txt".

Linking Elements to a Database

In simulations which require a great deal of input, it may be desirable (or even mandated) that the simulation model can access the various data sources directly to facilitate and ensure the quality of the data transfer.

One way to accomplish this in GoldSim is to import data from spreadsheets into a Spreadsheet element or into Lookup Tables or Time Series elements.

Read more: Linking a Lookup Table to a Spreadsheet (page 274); Importing Data into a Time Series from a Spreadsheet (page 194); Spreadsheet Elements (page 852).

GoldSim also provides a more powerful method. In particular, GoldSim data entry elements can be linked directly to an ODBC-compliant database.

After defining the linkage, you can then instruct GoldSim to download the data at any time. When it does this, GoldSim internally records the time and date at which the download occurred, along with other reference information retrieved from the database (e.g., document references), and this is stored with the model in the Run Log. This information is also displayed in the tool-tip for the linked element.

This allows you to confirm that the correct data were loaded into your model, and provides very strong and defensible quality control over your model input data.

The following elements can be downloaded from a database:

- Data elements;
- 1-D and 2-D Lookup Table elements; and
- Stochastic elements.

There are three steps involved in linking an element to a database:

1. Creating a database which is compatible with GoldSim;
2. Adding the database as a *data source* to your computer; and
3. Linking an element to the database and downloading the data.

Creating a Compatible Database

GoldSim can interact with three types of databases: 1) a Generic Database; 2) a Simple GoldSim Database; and 3) a Yucca Mountain Database. Each of these databases have specific formats and different capabilities, as outlined below:

Generic Database: Generic databases have a very simple format (consisting of a single table). They can download only to scalar, vector and matrix Data elements. They are the only type of database which can download to vector and matrix Data elements.

Simple GoldSim Database: Simple GoldSim databases are more structured than Generic databases. As such they can not only download to scalar, vector and matrix Data elements, but they can also download to Stochastic elements. The database is structured to allow you to store different versions of the same datum (e.g., associated with different dates). You specify from *within the database* which version is to be used when linked to GoldSim.

Yucca Mountain Database: The Yucca Mountain database format is the most complex of the three formats. They can download to scalar, vector and matrix Data elements, Stochastics, and 1-D and 2-D Table elements. The database is structured to allow you to store different versions of the same

datum (e.g., associated with different dates). You can specify from *within GoldSim* which version is to be used when linked to GoldSim.

The table below summarizes the key features of the three database formats:

	Download to Scalar Data Elements	Download to Vector and Matrix Data Elements	Download to Stochastics	Download to 1-D and 2-D Tables	Store Different Versions of the Same Datum
Generic	X	X			
Simple GoldSim	X	X	X		X
Yucca Mountain	X	X	X	X	X

The specific formats for these three types of databases are described in Appendix E.

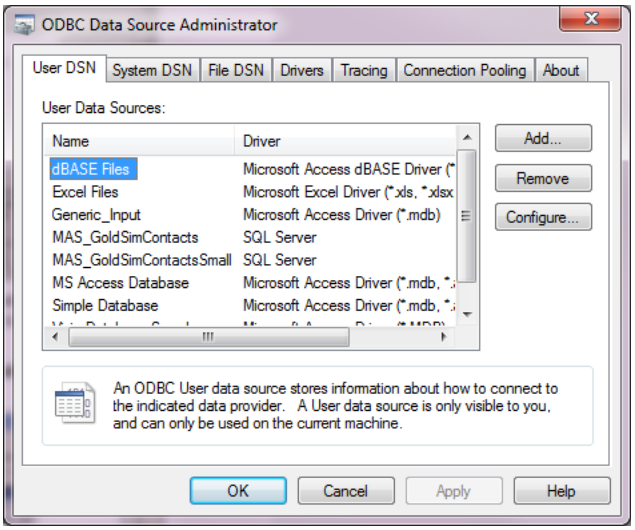
Adding Data Sources to Your Computer

Before using GoldSim's database features you must first identify the database(s) you need to access using the Windows Control Panel ODBC Data Sources (32bit) option. This allows you to define a name for each data source, and associate the data source with a specific database file.

For details in adding and configuring data sources, you should refer to your Microsoft Windows documentation. A brief overview is presented below.

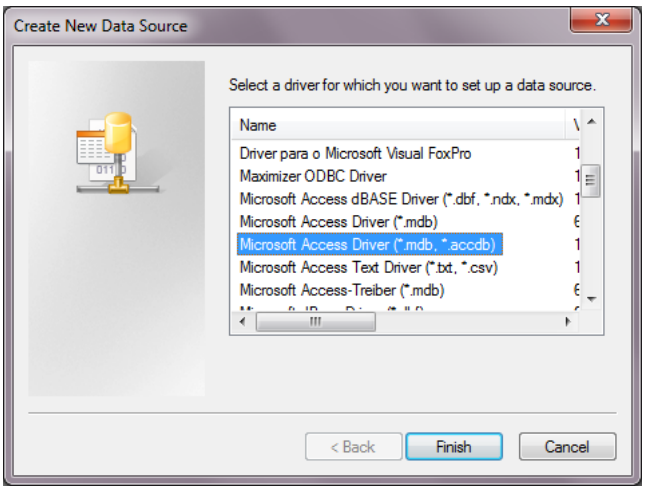
To add a data source to your computer, do the following (the instructions here are for Windows 7; they may differ slightly for other Windows versions):

1. From the Windows Control Panel, select **ODBC Data Sources**. A dialog similar to the one shown below will be displayed:



   You can define your database(s) as either 'User DSN' or 'System DSN', depending whether you want it to be private or public, respectively.

2. Add a data source by pressing the **Add…** button., which will display a dialog like this:



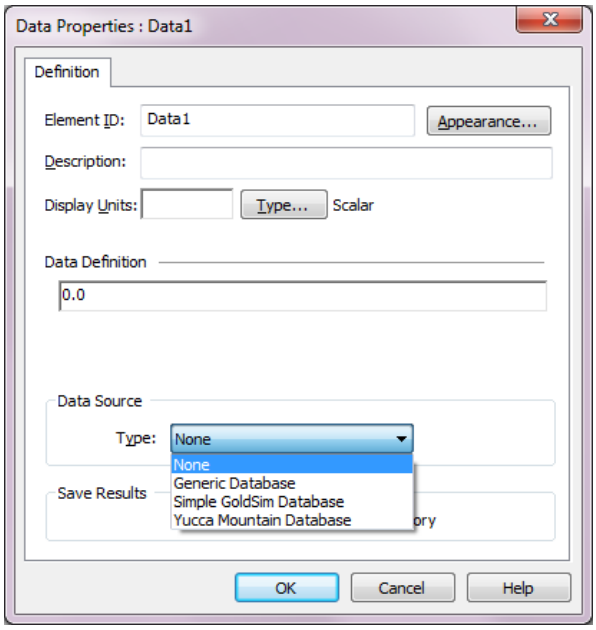
3. Select a driver with the same file extension as your database and press **Finish** (the driver selected above is for Microsoft Access 2003, 2007 and 2010). A dialog is displayed (the dialog will differ for each driver). You will be prompted to assign a name to the data source, and select the specific file (i.e., a database file)..

Once you have created the data source, you can reference it from within GoldSim.

Downloading Element Definitions from a Database

After defining the database and creating the data source, you must then link the database to specific elements and download the data.

In order to link a specific element to the database, you first select a database option from the **Data Source** field in the properties dialog for the element. This is typically found toward the bottom of the dialog:



You can also right-click on the element in the graphics pane or browser, and select **Connect to DataSource…**, which will display a menu for selecting the Data Source.

By default, the Data Source is "None" (i.e., the element is not linked to a database). To link to a database, you must select one of the database types from the drop-down list.

Note: Not all three database options are available in the drop-down list for all elements. For example, since a Stochastic cannot be linked to a Generic database, this option is not available for Stochastics in the **Database** tab.

When you do so, a **Database** tab is added to the element.

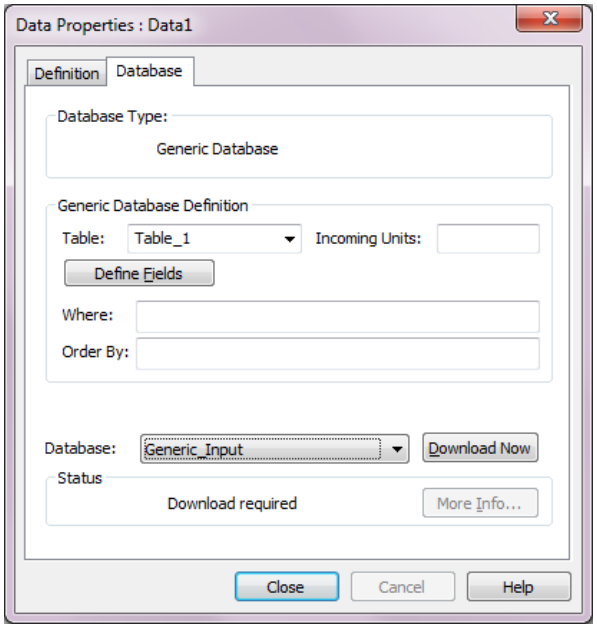
Three sample GoldSim files (Generic_DB.gsm, Simple_DB.gsm, and Yucca_DB.gsm) which illustrates the use of all three database types, are included in the General Examples/Database folder in your GoldSim directory. This folder also includes three sample databases referenced by these files (Generic_DB.accdb, Simple_DB. accdb, and Yucca_DB. accdb). In order to use the GoldSim files, you will need to add the databases as data sources to your computer.

Read more: Adding Data Sources to Your Computer (page 973).

Note: If you open a file which has elements linked to a database which GoldSim cannot find on your machine, it will prompt you to select from a list of databases that are registered on your machine. If none of these databases is appropriate, you must add a new data source to your computer. If you want GoldSim to check to see if it can find all of the databases referenced by elements in your model, press **Model | Database | Validate Databases**.

Downloading from a Generic Database

The following dialog is displayed when you specify a "Generic Database" for the **Data Source** for an element and click on the **Database** tab in the properties dialog for a Data element (the only element for which this type of data source can be specified):



As described in Appendix E, the Generic database format is very simple, and consists of a single table. To link to the database, you must specifically indicate the record and field you wish to access.

You link and download from a Generic database as follows:

1. Select the appropriate data source from the **Database** drop-down list (all defined data sources will be listed).
2. Select the appropriate table from the database from the **Table** drop-down list.
3. Press the **Define Fields** button in order to access a dialog for specifying which field(s) in the table contain the data. For scalar data, you will need to enter a single field (and this will be indicated at the top of the dialog). Vector and matrix data require multiple fields to be specified.
4. Specify which record (or for matrix data, records) in the table contain the data by specifying an appropriate condition in the **Where** input field.
5. Specify the incoming units of the data in the **Incoming Units** input field (using the appropriate GoldSim abbreviations for the units).
6. Press the **Download Now** button to download the data.

GoldSim notes whether the download was successful (and, if successful, displays the date and time of the download).

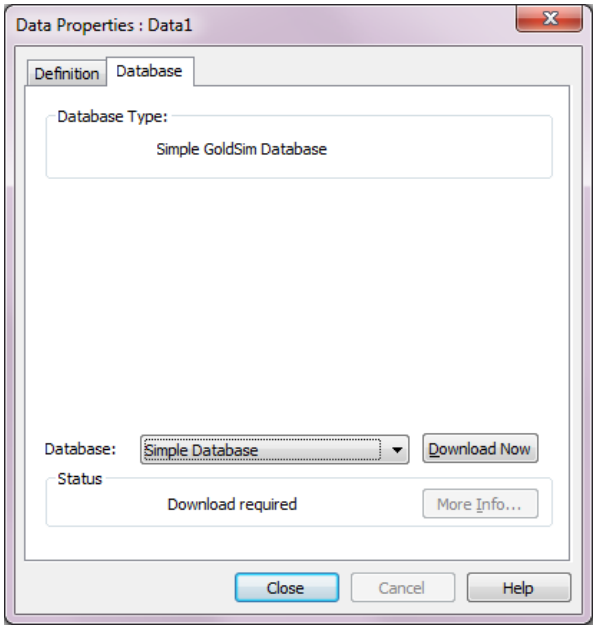
If you wish to download a vector, you must specify one record for each item in the vector. You identify the items of the vector using the **Where** input field. You must sort the records in the appropriate order using the **Order By** input field.

If you wish to download a matrix, you must specify one data field for each columns of the matrix (using the Define Fields dialog) and one record for each row of the matrix using the **Where** input field. The order of the fields (i.e., the columns of the matrix) is the order shown in the Define Fields dialog. You can use the **Move Up** button to modify the order. You must sort the records (i.e., the rows of the matrix) in the appropriate order using the **Order By** input field.

The file Generic_DB.gsm in the General Examples/Database folder of your GoldSim directory includes examples of how to download a vector and a matrix from a Generic database.

Downloading from a Simple GoldSim Database

The following dialog is displayed when you specify a "Simple GoldSim Database" for the **Data Source** for an element and click on the **Database** tab in the properties dialog for a Data or Stochastic element (the two elements for which this type of data source can be specified):



As described in Appendix E, the Simple GoldSim database has a specific structure in which a string in a field in one of the database tables must uniquely match the ID of the element you wish to link to.

When you link an element to a Simple GoldSim database, GoldSim locates the data to download within the database by using the element's ID and path. Unlike a Generic database, you do not need to manually indicate the record and field you wish to access from within GoldSim.

In addition, the units of data being downloaded are specified directly in the database, and need not be identified on the **Database** tab (although they must be consistent with the specified dimensions for the element as defined on the **Definition** tab).

Note: Particular care must be taken when importing Stochastics from a database, as different Stochastics could have different dimensions (e.g., a Binomial is always dimensionless; a Boolean is always a condition). If a Stochastic is defined in your model as a paticular type of distribution whose output is inconsistent with that of the distribution that you are trying to import from the database, GoldSim will display an error.

Note: Care must also be taken when importing Stochastics that represent temperatures. This is because temperatures have an absolute reference (i.e., absolute zero). *Differences* between temperatures are expressed in 'deg' units (Cdeg, Fdeg). This can lead to errors when importing Stochastics, since GoldSim assumes a single unit for the distribution parameters, while some types of distributions would require two different types of units. For example, a Normal distribution representing a temperature would require the Mean to be specifed in absolute units (e.g., C) and the Standard Deviation to be specified in difference units (e.g., Cdeg). If faced with this problem, there are several approaches for addressing this: 1) Use a distribution type where all parameters have the same units (e.g., triangular, uniform, cumulative); or 2) Specify and import the distributions as dimensionless and then apply a unit to the sampled value using an Expression element.

Read more: Dealing with Temperature Units (page 97).

Because a Simple GoldSim database is so highly structured, if the database is properly defined, linking and downloading from it requires only two steps:

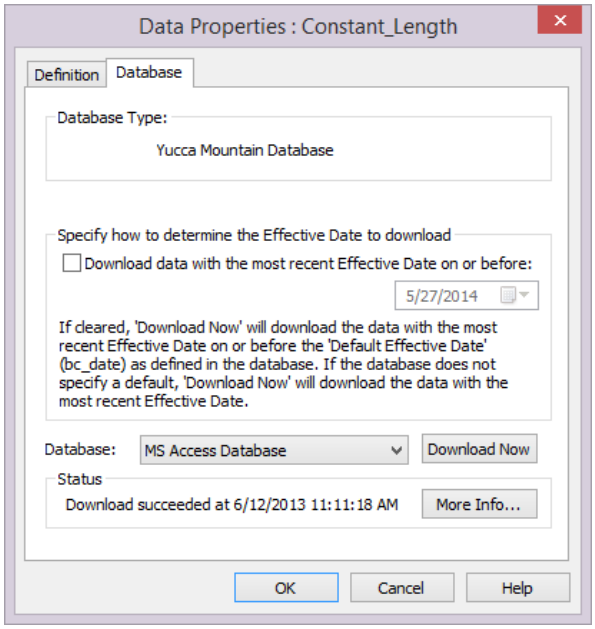
1. Select the appropriate data source from the **Database** drop-down list (all defined data sources will be listed).
2. Press the **Download Now** button to download the data.

GoldSim notes whether the download was successful (and, if successful, displays the date and time of the download).

The file Simple_DB.gsm in the General Examples/Database folder of your GoldSim directory includes examples of how to download data from a Simple GoldSim database.

Downloading from a Yucca Mountain Database

The following dialog is displayed when you specify a "Yucca Mountain Database" for the **Data Source** for an element and click on the **Database** tab in the properties dialog for Data, Stochastic or Lookup Table element (the three elements for which this type of data source can be specified):



As described in Appendix E, like the Simple GoldSim database, the Yucca Mountain database has a specific structure in which a string in a field in one of the database tables must uniquely match the ID of the element you wish to link to.

That is, when you link an element to a Yucca Mountain database, GoldSim locates the data to download within the database by using the element's ID. Unlike a Generic database, you do not need to manually indicate the record and field you wish to access from within GoldSim.

In addition, the units of data being downloaded are specified directly in the database, and need not be identified on the **Database** tab (although they must be consistent with the specified dimensions for the element as defined on the **Definition** tab).

Note: Particular care must be taken when importing Stochastics from a database, as different Stochastics could have different dimensions (e.g., a Binomial is always dimensionless; a Boolean is always a condition). If a Stochastic is defined in your model as a particular type of distribution whose output is inconsistent with that of the distribution that you are trying to import from the database, GoldSim will display an error.

Note: Care must also be taken when importing Stochastics that represent temperatures. This is because temperatures have an absolute reference (i.e., absolute zero). *Differences* between temperatures are expressed in 'deg' units (Cdeg, Fdeg). This can lead to errors when importing Stochastics, since GoldSim assumes a single unit for the distribution parameters, while some types of distributions would require two different types of units. For example, a Normal distribution representing a temperature would require the Mean to be specifed in absolute units (e.g., C) and the Standard Deviation to be specified in difference units (e.g., Cdeg). If faced with this problem, there are several approaches for addressing this: 1) Use a distribution type where all parameters have the same units (e.g., triangular, uniform, cumulative); or 2) Specify and import the distributions as dimensionless and then apply a unit to the sampled value using an Expression element.

Read more: Dealing with Temperature Units (page 97).

Because a Yucca Mountain database is so highly structured, if the database is properly defined, linking and downloading from it requires only two steps:

1. Select the appropriate data source from the **Database** drop-down list (all defined data sources will be listed).
2. Press the **Download Now** button to download the data.

GoldSim notes whether the download was successful (and, if successful, displays the date and time of the download).

The **More info…** button provides some additional information regarding a successful download (the effective date and the Run Log strings associated with the data).

As discussed in detail in Appendix E, for a Yucca Mountain database, you can create multiple records for a particular item of data, each with a different "Effective Date". This allows you to keep a record of how particular inputs to your model have been modified.

The checkbox allows you to download data associated with a specific Effective Date:

- If checked, you must specify a date, and **Download Now** will download the data with the most recent Effective Date on or before the specified date.
- If cleared, **Download Now** will download the data with the most recent Effective Date on or before the default Effective Date for the element, as specified in the GS_Parameter table of the database (described in Appendix E).
- If cleared and no default Effective Date is specified in the database for the element, **Download Now** will download the data with the most recent Effective Date

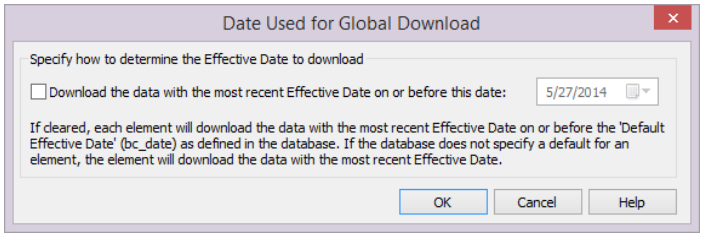
Note: As pointed out above, when you link an element to a Yucca Mountain database, GoldSim locates the data to download by using the element's ID alone (no path is specified in the database). Hence, in order for GoldSim to download the element's definition from the database, a record in the Element ID field of the database must match the GoldSim element ID exactly (although the comparison is not case-sensitive). This implies that you cannot have multiple elements with identical names (e.g. , in two localized containers) which are both being linked to a Yucca Mountain database.

The file Yucca_DB.gsm in the General Examples/Database folder of your GoldSim directory includes examples of how to download data from a Yucca Mountain database.

Globally Downloading Elements from a Database

In addition to downloading each element's definition individually from a database, you can also globally download all element definitions by pressing the Database Download button from the Standard toolbar (a box labeled DB), or selecting **Model | Database | Database Download** from the main menu.

If any of your elements are linked to a Yucca Mountain database, you are prompted to specify which Effective Date should be used for each of those elements:



The checkbox activates a Date field:

- If checked, you must specify a date, and each element will download the data with the most recent Effective Date on or before the specified date.
- If cleared, each element will download the data with the most recent Effective Date on or before the default Effective Date for the element, as specified in the GS_Parameter table of the database (described in Appendix E).

- If cleared and no default Effective Date is specified in the database for an element, each element will download the data with the most recent Effective Date.

If one or more of the downloads fails, a dialog will appear summarizing the elements for which the download failed.

Modifying Downloaded Data

While an element is linked to a database, you cannot access or edit the data. After you have downloaded data, however, you can choose to "uncouple" the element from the database by selecting the “No Database” option in the **Database** tab. When you do this, you will be able to view and edit the values downloaded from the database.

References

Embrechts, Paul., Lindskog, Filip and Alexander McNeil, 2001, "Modelling Dependence with Copulas and Applications to Risk Management," Department of Mathematics, Swiss Federal Institute of Technology, Zurich (http://www.defaultrisk.com/pp_corr_19.htm).

Dorey, Martyn, 2006, "Why Aren't Copulas Far More Popular?", Professional Investor, February 2006 (http://www.psolve.com/news/200602_Martyn%20Dorey%20article.pdf).

Appendix A: Introduction to Probabilistic Simulation

Our knowledge of the way things work, in society or in nature, comes trailing clouds of vagueness. Vast ills have followed a belief in certainty.

Kenneth Arrow, I Know a Hawk from a Handsaw

Appendix Overview

This appendix provides a very brief introduction to probabilistic simulation (the quantification and propagation of uncertainty). Because detailed discussion of this topic is well beyond the scope of this appendix, readers who are unfamiliar with this field are strongly encouraged to consult additional literature. A good introduction to the representation of uncertainty is provided by Finkel (1990) and a more detailed treatment is provided by Morgan and Henrion (1990). The basic elements of probability theory are discussed in Harr (1987) and more detailed discussions can be found in Benjamin and Cornell (1970) and Ang and Tang (1984).

In this Appendix

This appendix discusses the following:

- Types of Uncertainty
- Quantifying Uncertainty
- Propagating Uncertainty
- A Comparison of Probabilistic and Deterministic Analyses
- References

Types of Uncertainty

Many of the features, events and processes which control the behavior of a complex system will not be known or understood with certainty. Although there are a variety of ways to categorize the sources of this uncertainty, for the purpose of this discussion it is convenient to consider the following four types:

- Value (parameter) uncertainty: The uncertainty in the value of a particular parameter (e.g., a geotechnical property, or the development cost of a new product);
- Uncertainty regarding future events: The uncertainty in the ability to predict future perturbations of the system (e.g., a strike, an accident, or an earthquake).
- Conceptual model uncertainty: The uncertainty regarding the detailed understanding and representation of the processes controlling a particular system (e.g., the complex interactions controlling the flow rate in a river); and
- Numerical model uncertainty: The uncertainty introduced by approximations in the computational tool used to evaluate the system.

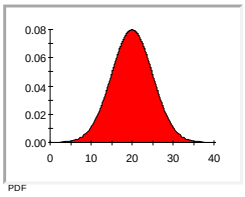
Incorporating these uncertainties into the predictions of system behavior is called *probabilistic analysis* or in some applications, *probabilistic performance assessment*. Probabilistic analysis consists of explicitly representing the uncertainty in the parameters, processes and events controlling the system and propagating this uncertainty through the system such that the uncertainty in the results (i.e., predicted future performance) can be quantified.

Quantifying Uncertainty

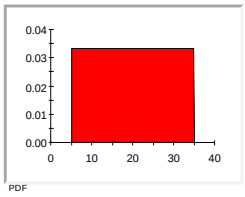
Understanding Probability Distributions

When uncertainty is quantified, it is expressed in terms of ***probability distributions***. A probability distribution is a mathematical representation of the relative likelihood of an uncertain variable having certain specific values.

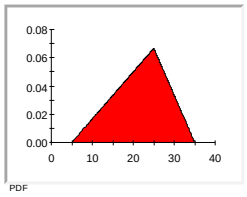
There are many types of probability distributions. Common distributions include the normal, uniform and triangular distributions, illustrated below:



Normal Distribution



Uniform Distribution



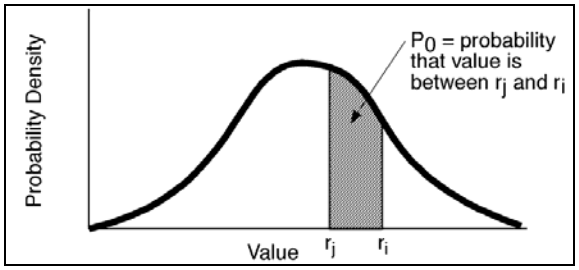
Triangular Distribution

All distribution types use a set of *arguments* to specify the relative likelihood for each possible value. For example, the normal distribution uses a ***mean*** and a ***standard deviation*** as its arguments. The mean defines the value around which the bell curve will be centered, and the standard deviation defines the spread of values around the mean. The arguments for a uniform distribution are a minimum and a maximum value. The arguments for a triangular distribution are a minimum value, a most likely value, and a maximum value.

The nature of an uncertain parameter, and hence the form of the associated probability distribution, can be either *discrete* or *continuous*. Discrete distributions have a limited (discrete) number of possible values (e.g., 0 or 1; yes

or no; 10, 20, or 30). Continuous distributions have an infinite number of possible values (e.g., the normal, uniform and triangular distributions shown above are continuous). Good overviews of commonly applied probability distributions are provided by Morgan and Henrion (1990) and Stephens et al. (1993).

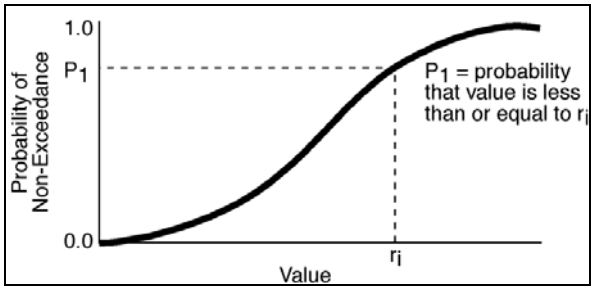
There are a number of ways in which probability distributions can be graphically displayed. The simplest way is to express the distribution in terms of a ***probability density function*** (PDF), which is how the three distributions shown above are displayed. In simple terms, this plots the relative likelihood of the various possible values, and is illustrated schematically below:



Note that the "height" of the PDF for any given value is *not* a direct measurement of the probability. Rather, it represents the *probability density*, such that integrating under the PDF between any two points results in the probability of the actual value being between those two points.

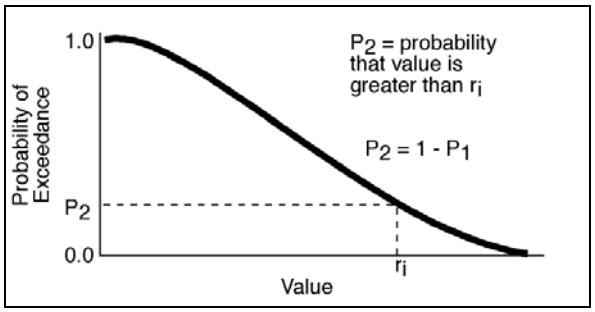
Note: Discrete distributions are described mathematically using probability mass functions (pmf), rather than probability density functions. Probability mass functions specify actual probabilities for given values, rather than probability densities.

An alternative manner of representing the same information contained in a PDF is the ***cumulative distribution function*** (CDF). This is formed by integrating over the PDF (such that the slope of the CDF at any point equals the height of the PDF at that point). For any point on the horizontal axis **r**, the CDF shows the cumulative probability that the actual value will be less than or equal to **r**. That is, as shown below, a particular point, say $\mathbf{[r_i , P_1]}$, on the CDF is interpreted as follows: $\mathbf{P_1}$ is the probability that the actual value is less than or equal to $\mathbf{r_i}$.



By definition, the total area under the PDF must integrate to 1.0, and the CDF therefore ranges from 0.0 to 1.0.

A third manner of presenting this information is the ***complementary cumulative distribution function*** (CCDF). The CCDF is illustrated schematically below:



A particular point, say **[r_i, P_2]**, on the CCDF is interpreted as follows: **P_2** is the probability that the actual value is greater than **r_i** . Note that the CCDF is simply the complement of the CDF; that is, **P_2** is equal to 1 - **P_1**.

Probability distributions are often described using *quantiles* or *percentiles* of the CDF. Percentiles of a distribution divide the total frequency of occurrence into hundredths. For example, the 90th percentile is that value of the parameter below which 90% of the distribution lies. The 50th percentile is referred to as the ***median***.

Characterizing Distributions

Probability distributions can be characterized by their *moments*. The first moment is referred to as the ***mean*** or ***expected value***, and is typically denoted as μ. For a continuous distribution, it is computed as follows:

$$\mu = \int x\, f(x)\, dx$$

where f(x) is the probability density function (PDF) of the variable. For a discrete distribution, it is computed as:

$$\mu = \sum_{i=1}^{N} x_i\, p(x_i)$$

in which $p(x_i)$ is the probability of x_i, and N is the total number of discrete values in the distribution.

Additional moments of a distribution can also be computed. The nth moment of a continuous distribution is computed as follows:

$$\mu_n = \int (x - \mu)^n \, f(x) \, dx$$

For a discrete distribution, the nth moment is computed as:

$$\mu_n = \sum_{i=1}^{N} (x_i - \mu)^n \, p(x_i)$$

The second moment is referred to as the ***variance***, and is typically denoted as σ^2. The square root of the variance, σ, is referred to as the ***standard deviation***. The variance and the standard deviation reflect the amount of spread or dispersion in the distribution. The ratio of the standard deviation to the mean provides a dimensionless measure of the spread, and is referred to as the ***coefficient of variation***.

The ***skewness*** is a dimensionless number computed based on the third moment:

$$\text{skewness} = \frac{\mu_3}{\sigma^3}$$

The skewness indicates the symmetry of the distribution. A normal distribution (which is perfectly symmetric) has a skewness of zero. A positive skewness indicates a shift to the right (and example is the log-normal distribution). A negative skewness indicates a shift to the left.

The ***kurtosis*** is a dimensionless number computed based on the fourth moment:

$$\text{kurtosis} = \frac{\mu_4}{\sigma^4}$$

The kurtosis is a measure of how "fat" a distribution is, measured relative to a normal distribution with the same standard deviation. A normal distribution has a kurtosis of zero. A positive kurtosis indicates that the distribution is more "peaky" than a normal distribution. A negative kurtosis indicates that the distribution is "flatter" than a normal distribution.

Specifying Probability Distributions

Given the fact that probability distributions represent the means by which uncertainty can be quantified, the task of quantifying uncertainty then becomes a matter of assigning the appropriate distributional forms and arguments to the uncertain aspects of the system. Occasionally, probability distributions can be defined by fitting distributions to data collected from experiments or other data collection efforts. For example, if one could determine that the uncertainty in a particular parameter was due primarily to random measurement errors, one might simply attempt to fit an appropriate distribution to the available data.

Most frequently, however, such an approach is not possible, and probability distributions must be based on *subjective assessments* (Bonano et al., 1989; Roberds, 1990; Kotra et al., 1996). Subjective assessments are opinions and judgments about probabilities, based on experience and/or knowledge in a specific area, which are consistent with available information. The process of developing these assessments is sometimes referred to as *expert elicitation*. Subjectively derived probability distributions can represent the opinions of individuals or of groups. There are a variety of methods for developing subjective probability assessments, ranging from simple informal techniques to complex and time-consuming formal methods. It is beyond the scope of this document to discuss these methods. Roberds (1990), however, provides an

overview, and includes a list of references. Morgan and Henrion (1990) also provide a good discussion on the topic.

A key part of all of the various approaches for developing subjective probability assessments is a methodology for developing (and justifying) an appropriate probability distribution for a parameter in a manner that is logically and mathematically consistent with the level of available information. Discussions on the applicability of various distribution types are provided by Harr (1987, Section 2.5), Stephens et al. (1993), and Seiler and Alvarez (1996). Note that methodologies (Bayesian updating) also exist for updating an existing probability distribution when new information becomes available (e.g., Dakins, et al., 1996).

Correlated Distributions

Frequently, parameters describing a system will be *correlated* (inter-dependent) to some extent. For example, if one were to plot frequency distributions of the height and the weight of the people in an office, there would likely be some degree of positive correlation between the two: taller people would generally also be heavier (although this correlation would not be perfect).

The degree of correlation can be measured using a ***correlation coefficient***, which varies between 1 and -1. A correlation coefficient of 1 or -1 indicates perfect positive or negative correlation, respectively. A positive correlation indicates that the parameters increase or decrease together. A negative correlation indicates that increasing one parameter decreases the other. A correlation coefficient of 0 indicates no correlation (the parameters are apparently independent of each other). Correlation coefficients can be computed based on the actual values of the parameters (which measures linear relationships) or the rank-order of the values of the parameters (which can be used to measure non-linear relationships).

One way to express correlations in a system is to directly specify the correlation coefficients between various model parameters. In practice, however, assessing and quantifying correlations in this manner is difficult. Oftentimes, a more practical way of representing correlations is to explicitly model the cause of the dependency. That is, the analyst adds detail to the model such that the underlying functional relationship causing the correlation is directly represented.

For example, one might be uncertain regarding the solubility of two contaminants in water, while knowing that the solubilities tend to be correlated. If the main source of this uncertainty was actually uncertainty in pH conditions, and the solubility of each contaminant was expressed as a function of pH, the distributions of the two solubilities would then be explicitly correlated. If both solubilities increased or decreased with increasing pH, the correlation would be positive. If one decreased while one increased, the correlation would be negative.

Ignoring correlations, particularly if they are very strong (i.e., the absolute value of the correlation coefficient is close to 1) can lead to physically unrealistic simulations. In the above example, if the solubilities of the two contaminants were positively correlated (e.g., due to a pH dependence), it would be physically inconsistent for one contaminant's solubility to be selected from the high end of its possible range while the other's was selected from the low end of its possible range. Hence, when defining probability distributions, it is critical that the analyst determine whether correlations need to be represented.

Variability and Ignorance

When quantifying the uncertainty in a system, there are two fundamental causes of uncertainty which are important to distinguish: 1) that due to inherent variability; and 2) that due to ignorance or lack of knowledge. IAEA (1989) refers to the former as "Type A uncertainty" and the latter as "Type B

uncertainty". These are also sometimes referred to as *aleatory* and *epistemic* uncertainty, respectively.

Type A uncertainty results from the fact that many parameters are inherently *variable* over space and/or time. Examples include the height of trees in a forest or the flow rate in a river. If one were to ask "What is the height of the forest?" or "What is the flow rate in the river at point A?", even if we had perfect information (i.e., complete knowledge), the answer to these questions would be distributions (in space and time, respectively) as opposed to single values. Variability in a parameter can be expressed using *frequency distributions*. A frequency distribution displays the relative frequency of a particular value versus the value. For example, one could sample the flow rate of a river once a day for a year, and plot a frequency distribution of the daily flow rate (the x-axis being the flow rate, and the y-axis being the frequency of the observation over the year).

If on the other hand, one were to ask "What is the average height of trees in the forest?" or "What is the peak flow rate in the river at point A?", the answers to these questions would be single values. In practice, of course, in both of these cases, we often could not answer these questions precisely due to Type B uncertainty: we lack sufficient information or knowledge about the system to answer the questions with absolute certainty.

If an answer consists of a single value about which we are uncertain due to a lack of knowledge (Type B uncertainty), the quantity can be represented by a probability distribution which quantifies the degree of uncertainty. If an answer consists of a distribution due to inherent variability (Type A uncertainty) about which we are uncertain due to a lack of knowledge (Type B uncertainty), the quantity can be represented by a frequency distribution whose arguments themselves (e.g., mean and standard deviation) are probability distributions.

Parameters which are both uncertain and inherently variable are not uncommon. For example, in considering the side effects of a new drug, there will likely be inherent variability in the sensitivity to the drug among the population (e.g., children may be more sensitive than adults), and there may also be poor scientific understanding as to the actual sensitivity of any particular population group to the drug.

Whenever possible, it is usually preferable to explicitly distinguish variability from ignorance. In the above example, this could be accomplished to a large extent by defining separate sensitivity factors for each of a number of sub-populations (e.g., male adults, female adults, children). Doing so allows the analyst to determine to what degree the uncertainty in the key input parameters (and hence the uncertainty in the impacts) can be reduced: uncertainty due to variability is inherently irreducible (and can only be represented statistically), but uncertainty due to ignorance could potentially be reduced by collecting more data and carrying out further research.

The key point here is that the analyst should be careful to distinguish between these two types of uncertainty in order to determine to what degree each needs to be represented in the simulation model.

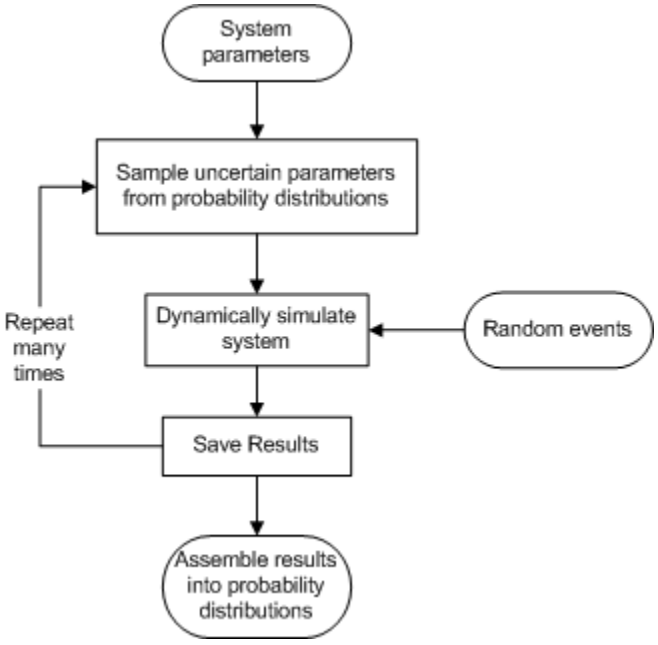
Propagating Uncertainty

If the inputs describing a system are uncertain, the prediction of the future performance of the system is necessarily uncertain. That is, the result of any analysis based on inputs represented by probability distributions is itself a probability distribution.

In order to compute the probability distribution of predicted performance, it is necessary to *propagate* (translate) the input uncertainties into uncertainties in the results. A variety of methods exist for propagating uncertainty. Morgan and Henrion (1990) provide a relatively detailed discussion on the various methods.

One common technique for propagating the uncertainty in the various aspects of a system to the predicted performance (and the one used by GoldSim) is ***Monte Carlo simulation***. In Monte Carlo simulation, the entire system is simulated a large number (e.g., 1000) of times. Each simulation is equally likely, and is referred to as a ***realization*** of the system. For each realization, all of the uncertain parameters are sampled (i.e., a single random value is selected from the specified distribution describing each parameter). The system is then simulated through time (given the particular set of input parameters) such that the performance of the system can be computed.

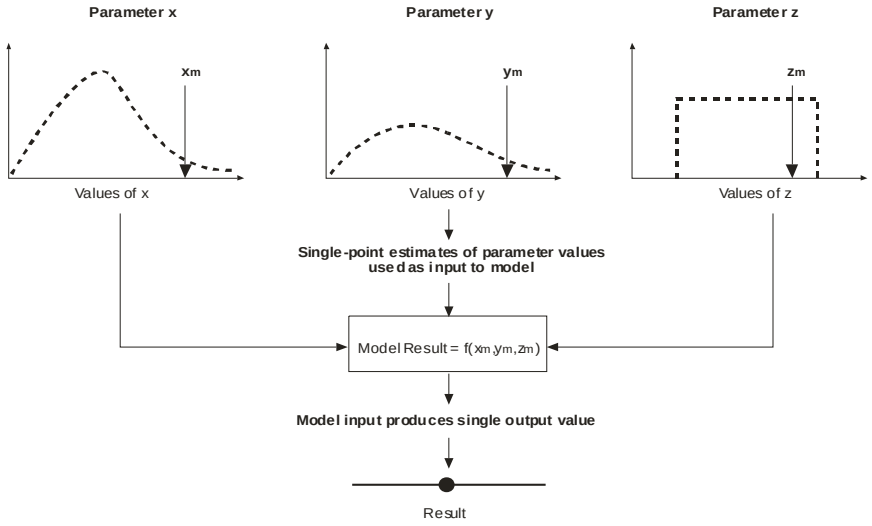
This results in a large number of separate and independent results, each representing a possible "future" for the system (i.e., one possible path the system may follow through time). The results of the independent system realizations are assembled into probability distributions of possible outcomes. A schematic of the Monte Carlo method is shown below:



A Comparison of Probabilistic and Deterministic Simulation Approaches

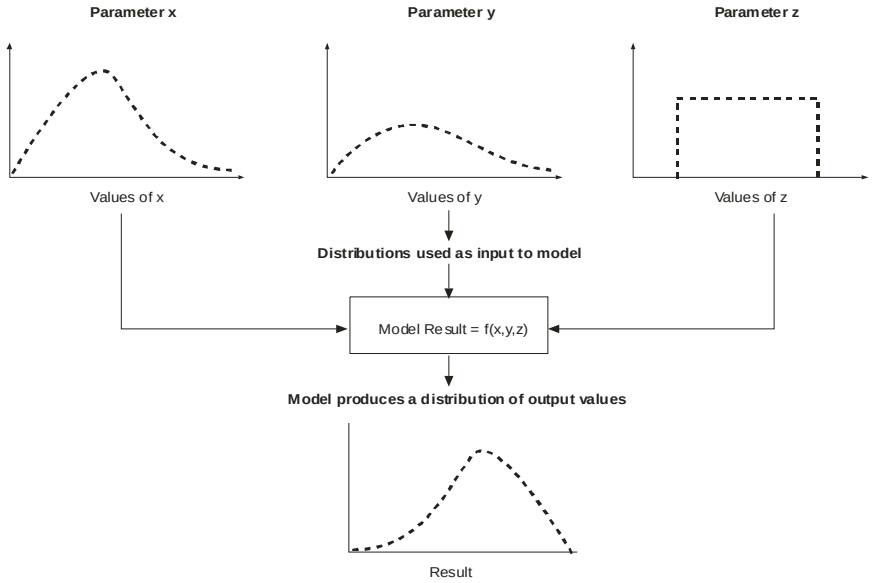
Having described the basics of probabilistic analysis, it is worthwhile to conclude this appendix with a comparison of probabilistic and ***deterministic*** approaches to simulation, and a discussion of why GoldSim was designed to specifically facilitate both of these approaches.

The figure below shows a schematic representation of a deterministic modeling approach:



In the deterministic approach, the analyst, although he/she may implicitly recognize the uncertainty in the various input parameters, selects single values for each parameter. Typically, these are selected to be "best estimates" or sometimes "worst case estimates". These inputs are evaluated using a simulation model, which then outputs a single result, which presumably represents a "best estimate" or "worst case estimate".

The figure below shows a similar schematic representation of a probabilistic modeling approach:



In this case the analyst explicitly represents the input parameters as probability distributions, and propagates the uncertainty through to the result (e.g., using the Monte Carlo method), such that the result itself is also a probability distribution.

One advantage to deterministic analyses is that they can typically incorporate more detailed components than probabilistic analyses due to computational considerations (since complex probabilistic analyses generally require time-consuming simulation of multiple realizations of the system).

Deterministic analyses, however, have a number of disadvantages:

- *"Worst case" deterministic simulations can be extremely misleading*. Worst case simulations of a system may be grossly conservative and therefore completely unrealistic (i.e., they typically have an extremely low probability of actually representing the future behavior of the system). Moreover, it is not possible in a deterministic simulation to quantify how conservative a "worst case" simulation actually is. Using a highly improbable simulation to guide policy making (e.g., "is the design safe?") is likely to result in poor decisions.
- *"Best estimate" deterministic simulations are often difficult to defend*. Because of the inherent uncertainty in most input parameters, defending "best estimate" parameters is often very difficult. In a confrontational environment, "best estimate" analyses will typically evolve into "worst case" analyses.
- *Deterministic analyses do not lend themselves directly to detailed uncertainty and sensitivity studies*. In order to carry out uncertainty and sensitivity analysis of deterministic simulations, it is usually necessary to carry out a series of separate simulations in which various parameters are varied. This is time-consuming and typically results only in a limited analysis of sensitivity and uncertainty.

These disadvantages do not exist for probabilistic analyses. Rather than facing the difficulties of defining worst case or best estimate inputs, probabilistic analyses attempt to explicitly represent the full range of possible values. The probabilistic approach embodied within GoldSim acknowledges the fact that for many complex systems, predictions are inherently uncertain and should always be presented as such. Probabilistic analysis provides a means to present this uncertainty in a quantitative manner.

Moreover, the output of probabilistic analyses can be used to directly determine parameter sensitivity. Because the output of probabilistic simulations consists of multiple sets of input parameters and corresponding results, the sensitivity of results to various input parameters can be directly determined. The fact that probabilistic analyses lend themselves directly to evaluation of parameter sensitivity is one of the most powerful aspects of this approach, allowing such tools to be used to aid decision-making.

There are, however, some potential disadvantages to probabilistic analyses that should also be noted:

- *Probabilistic analyses may be perceived as unnecessarily complex, or unrealistic*. Although this sentiment is gradually becoming less prevalent as probabilistic analyses become more common, it cannot be ignored. It is therefore important to develop and present probabilistic analyses in a manner that is straightforward and transparent. In fact, GoldSim was specifically intended to minimize this concern.
- *The process of developing input for a probabilistic analysis can sometimes degenerate into futile debates about the "true" probability*

distributions. This concern can typically be addressed by simply repeating the probabilistic analysis using alternative distributions. If the results are similar, then there is not necessity to pursue the "true" distributions further.

- *The public (courts, media, etc.) typically does not fully understand probabilistic analyses and may be suspicious of it*. This may improve as such analyses become more prevalent and the public is educated, but is always likely to be a problem. As a result, complementary deterministic simulations will always be required in order to illustrate the performance of the system under a specific set of conditions (e.g., "expected" or "most likely" conditions).

As this last point illustrates, it is important to understand that use of a probabilistic analysis does not preclude the use of deterministic analysis. In fact, deterministic analyses of various system components are often essential in order to provide input to probabilistic analyses. The key point is that for many systems, deterministic analyses *alone* can have significant disadvantages and in these cases, they should be complemented by probabilistic analyses.

References

The references cited in this appendix are listed below.

Ang, A. H-S. and W.H. Tang, 1984, Probability Concepts in Engineering Planning and Design, Volume II: Decision, Risk, and Reliability, John Wiley & Sons, New York.

Bonano, E.J., S.C. Hora, R.L. Keaney and C. von Winterfeldt, 1989, Elicitation and Use of Expert Judgment in Performance Assessment for High-Level Radioactive Waste Repositories, Sandia Report SAND89-1821, Sandia National Laboratories.

Benjamin, J.R. and C.A. Cornell, 1970, Probability, Statistics, and Decision for Civil Engineers, McGraw-Hill, New York.

Dakins, M.E., J.E. Toll, M.J. Small and K.P. Brand, 1996, *Risk-Based Environmental Remediation: Bayesian Monte Carlo Analysis and the Expected Value of Sample Information*, Risk Analysis, Vol. 16, No. 1, pp. 67-79.

Finkel, A., 1990, Confronting Uncertainty in Risk Management: A Guide for Decision-Makers, Center for Risk Management, Resources for the Future, Washington, D.C.

Harr, M.E., 1987, Reliability-Based Design in Civil Engineering, McGraw-Hill, New York.

IAEA, 1989, Evaluating the Reliability of Predictions Made Using Environmental Transfer Models, IAEA Safety Series No. 100, International Atomic Energy Agency, Vienna.

Kotra, J.P., M.P. Lee, N.A. Eisenberg, and A.R. DeWispelare, 1996, *Branch Technical Position on the Use of Expert Elicitation in the High-Level Radioactive Waste Program*, Draft manuscript, February 1996, U.S. Nuclear Regulatory Commission.

Morgan, M.G. and M. Henrion, 1990, Uncertainty, Cambridge University Press, New York.

Roberds, W.J., 1990, *Methods for Developing Defensible Subjective Probability Assessments*, Transportation Research Record, No. 1288,

Transportation Research Board, National Research Council, Washington, D.C., January 1990.

Seiler, F.A and J.L. Alvarez, 1996, *On the Selection of Distributions for Stochastic Variables*, Risk Analysis, Vol. 16, No. 1, pp. 5-18.

Stephens, M.E., B.W. Goodwin and T.H. Andres, 1993, *Deriving Parameter Probability Density Functions*, Reliability Engineering and System Safety, Vol. 42, pp. 271-291.

Appendix B: Probabilistic Simulation Details

Clever liars give details, but the cleverest don't.

Anonymous

Appendix Overview

This appendix provides the mathematical details of how GoldSim represents and propagates uncertainty, and the manner in which it constructs and displays probability distributions of computed results. While someone who is not familiar with the mathematics of probabilistic simulation should find this appendix informative and occasionally useful, most users need not be concerned with these details. Hence, this appendix is primarily intended for the serious analyst who is quite familiar with the mathematics of probabilistic simulation and wishes to understand the specific algorithms employed by GoldSim.

In this Appendix

This appendix discusses the following:

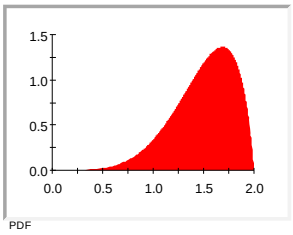
- Mathematical Representation of Probability Distributions
- Correlation Algorithms
- Sampling Techniques
- Representing Random (Poisson) Events
- Computing and Displaying Result Distributions
- Computing Sensitivity Analysis Measures
- References

Mathematical Representation of Probability Distributions

Distributional Forms

The arguments, probability density (or mass) function (pdf or pmf), cumulative distribution function (cdf), and the mean and variance for each of the probability distributions available within GoldSim are presented below.

Beta Distribution



The beta distribution for a parameter is specified by a minimum value (a), a maximum value (b), and two shape parameters denoted S and T. The beta distribution represents the distribution of the underlying probability of success for a binomial sample, where S represents the observed number of successes in a binomial trial of T total draws.

Alternative formulations of the beta distribution use parameters α and β, or α_1 and α_2, where $S = \alpha = \alpha_1$ and $(T-S) = \beta = \alpha_2$.

Frequently the Beta distribution is also defined in terms of a minimum, maximum, mean, and standard deviation. The shape parameters are then computed from these statistics.

The beta distribution has many variations controlled by the shape parameters. It is always limited to the interval (a,b). Within (a,b), however, a variety of distribution forms are possible (e.g., the distribution can be configured to behave exponentially, positively or negatively skewed, and symmetrically). The distribution form obtained by different S and T values is predictable for a skilled user.

pdf: $$f(x) = \frac{1}{B(b-a)^{T-1}}(x-a)^{S-1}(b-x)^{T-S-1}$$

where: $$B = \frac{\Gamma(S)\Gamma(T-S)}{\Gamma(T)}$$

$$\Gamma(k) = \int_0^\infty e^{-u}u^{k-1}du$$

cdf: No closed form

mean: $$\mu = a + \frac{S}{T}(b-a)$$

variance: $$\sigma^2 = (b-a)^2\frac{S(T-S)}{T^2(T+1)}$$

Note that within GoldSim, there are three ways to define a Beta distribution. You can choose to specify S and T (Beta Distribution). Alternatively, you can specify a mean, standard deviation, minimum and maximum, as defined above (Generalized Beta Distribution). In this case, GoldSim limits the standard deviations that can be specified as follows:

$$\sigma^* <= 0.6\sqrt{\mu^*(1-\mu^*)}$$

where $\mu^* = \frac{\mu - a}{b - a}$, $\sigma^* = \frac{\sigma}{b - a}$

This constraint ensures that the distribution has a single peak and that it does not have a discrete probability mass at either end of its range.

Finally, you can specify a minimum (a), maximum (b) and most likely value (c) (BetaPERT distribution). In this case, GoldSim assumes shape parameters are as follows:

$$\alpha = 1 + 4c_n$$

$$\beta = 5 - 4c_n$$

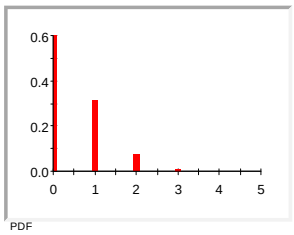
where

$$c_n = \frac{c - a}{b - a}$$

Note that in this case, the most likely value specified by the user is not mathematically the most likely value (but is a very close approximation to it).

Note that if the BetaPert is defined using the 10th and 90th percentile (instead of a minimum and a maximum) the minimum and maximum are estimated through iteration.

Binomial Distribution



The binomial distribution is a discrete distribution specified by a batch size (n) and a probability of occurrence (p). This distribution can be used to model the number of parts that failed from a given set of parts, where n is the number of parts and p is the probability of the part failing.

pmf: $P(x) = \binom{n}{x} p^x (1-p)^{n-x}$ x = 0, 1, 2, 3...

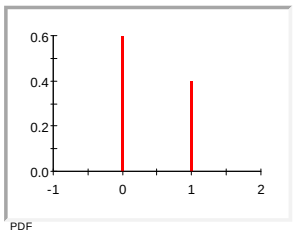
where: $\binom{n}{x} = \frac{n!}{x!(n-x)!}$

cdf: $F(x) = \sum_{i=0}^{x} \binom{n}{i} p^i (1-p)^{n-i}$

mean: np

variance: np(1-p)

Boolean Distribution



The Boolean (or logical) distribution requires a single input: the probability of being true, p. The distribution takes on one of two values: False (0) or True (1).

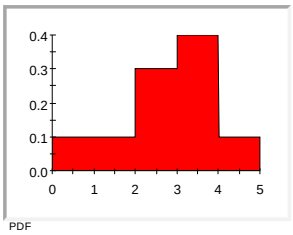
pmf: $P(x) = \begin{cases} 1-p & x=0 \\ p & x=1 \end{cases}$

cdf: $F(x) = \begin{cases} 1-p & x=0 \\ 1 & x=1 \end{cases}$

mean: $\mu = p$

variance: $\sigma^2 = p(1 - p)$

Cumulative Distribution



The cumulative distribution enables the user to input a piece-wise linear cumulative distribution function by simply specifying value (x_i) and cumulative probability (p_i) pairs.

GoldSim allows input of an unlimited number of pairs, x_i, p_i. In order to conform to a cumulative distribution function, it is a requirement that the first probability equal 0 and the last equal 1. The associated values, denoted x_0 and x_n, respectively, define the minimum value and maximum value of the distribution.

pdf:

$$f(x) = \begin{cases} 0 & x \le x_0 \text{ or } x \ge x_n \\ \dfrac{p_{i+1} - p_i}{x_{i+1} - x_i} & x_i \le x \le x_{i+1} \end{cases}$$

cdf:

$$F(x) = \begin{cases} 0 & x \le x_0 \\ p_i + (p_{i+1} - p_i)\dfrac{x - x_i}{x_{i+1} - x_i} & x_i \le x \le x_{i+1} \\ 1 & x \ge x_n \end{cases}$$

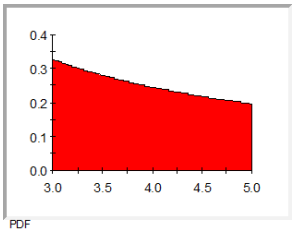
mean:

$$\mu \cong \sum_{i=1}^{n} x_i f(x_i)$$

variance:

$$\sigma^2 \cong \sum_{i=1}^{n} x_i^2 f(x_i) - \mu^2$$

Log- Cumulative Distribution



The log-cumulative distribution enables the user to input a piece-wise logarithmic cumulative distribution function by simply specifying value (x_i) and cumulative probability (p_i) pairs. Whereas in a cumulative distribution, the density between values is constant (i.e., the distribution between values is uniform), in a log-cumulative, the density of the *log* of the value is constant (i.e., the distribution between values is log-uniform).

GoldSim allows input of an unlimited number of pairs, x_i, p_i. In order to conform to a cumulative distribution function, it is a requirement that the first probability equal 0 and the last equal 1. The associated values, denoted x_0 and x_n, respectively, define the minimum value and maximum value of the distribution. Also, all values must be positive.

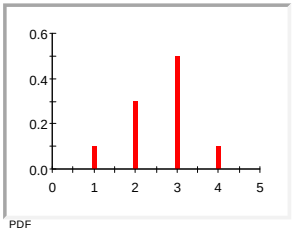
pdf:
$$f(x) = \begin{cases} 0 & x \le x_0 \text{ or } x \ge x_n \\ \dfrac{p_{i+1} - p_i}{x \ln(x_{i+1} / x_i)} & x_i \le x \le x_{i+1} \end{cases}$$

cdf:
$$F(x) = \begin{cases} 0 & x \le x_0 \\ p_i \dfrac{\ln(x_{i+1} / x)}{\ln(x_{i+1} / x_i)} + p_{i+1} \dfrac{\ln(x / x_i)}{\ln(x_{i+1} / x_i)} & x_i \le x \le x_{i+1} \\ 1 & x \ge x_n \end{cases}$$

mean:
$$\mu \cong \sum_{i=1}^{n} \frac{(p_{i+1} - p_i)(x_{i+1} - x_i)}{\ln(x_{i+1} / x_i)}$$

variance:
$$\sigma^2 \cong \sum_{i=1}^{n} \frac{(p_{i+1} - p_i)(x_{i+1}^{\ 2} - x_i^{\ 2})}{2\ln(x_{i+1} / x_i)}$$

Discrete Distribution



The discrete distribution enables the user to directly input a probability mass function for a discrete parameter. Each discrete value, x_i, that may be assigned to the parameter, has an associated probability, p_i, indicating its likelihood to occur. To conform to the requirements of a probability mass function, the sum of the probabilities, p_i , must equal 1. The discrete distribution is commonly used for situations with a small number of possible outcomes, such as "flag" variables used to indicate the occurrence of certain conditions.

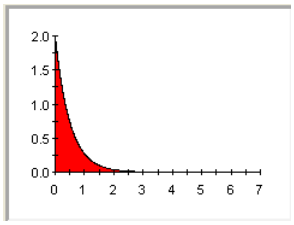
pmf:
$$P(x_i) = p_i \qquad x = x_i$$

cdf:
$$F(x_i) = \sum_{j=1}^{i \ge j} p_j$$

mean:
$$\mu \cong \sum_{i=1}^{n} x_i p_i$$

variance:
$$\sigma^2 \cong \sum_{i=1}^{n} x_i^2 p_i - \mu^2$$

Exponential Distribution



The Exponential distribution is a continuous distribution specified by a mean value (μ) which must be positive. This distribution is typically used to model the time required to complete a task or achieve a milestone.

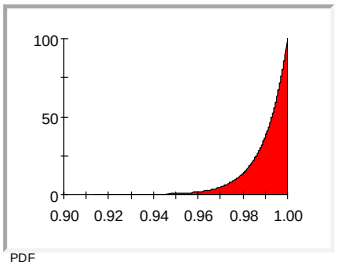
pdf: $$f(x)=\begin{cases}\frac{1}{\mu}e^{-\frac{x}{\mu}} & if \quad x \geq 0 \\ 0 & otherwise\end{cases}$$

cdf: $$F(x)=\begin{cases}1-e^{-\frac{x}{\mu}} & if \quad x \geq 0 \\ 0 & otherwise\end{cases}$$

mean: μ

variance: μ^2

Extreme Probability Distribution



The Extreme Probability distribution provides the expected extreme probability level of a uniform distribution given a specific number of samples. Both the Minimum and Maximum Extreme Probability Distributions are equivalent to the Beta distribution with the following parameters:

	Maximum	Minimum
S	Number of samples	1
(T-S)	1	Number of Samples

pdf: $$f(x) = \frac{1}{B(b-a)^{T-1}}(x-a)^{S-1}(b-x)^{T-S-1}$$

where: $$B = \frac{\Gamma(S)\Gamma(T-S)}{\Gamma(T)}$$

$$\Gamma(k) = \int_0^\infty e^{-u}u^{k-1}du$$

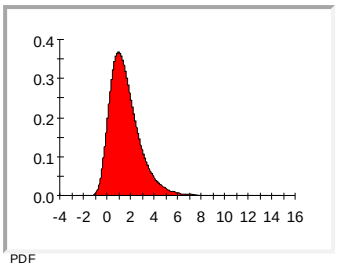
cdf: No closed form

mean: $$\mu = a + \frac{S}{T}(b-a)$$

variance: $$\sigma^2 = (b-a)^2\frac{S(T-S)}{T^2(T+1)}$$

Extreme Value Distribution

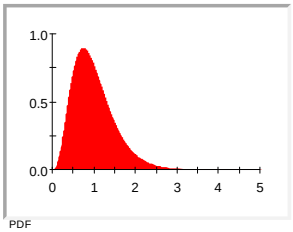
The Extreme Value distribution (also known as the Gumbel distribution) is used to represent the maximum or minimum expected value of a variable. It is specified with the mode (m) and a scale parameter (s) that must be positive.



	Maximum	Minimum
pdf	$f(x) = \frac{z}{s}e^{-z}$ where $z = e^{\frac{-(x-m)}{s}}$	$f(x) = \frac{z}{s}e^{-z}$ where $z = e^{\frac{(x-m)}{s}}$
cdf	$F(x) = e^{-z}$	$F(x) = 1 - e^{-z}$
mean	$\mu = m + 0.57722s$	$\mu = m - 0.57722s$
Variance	$\sigma^2 = \frac{(s\pi)^2}{6}$	$\sigma^2 = \frac{(s\pi)^2}{6}$

Gamma Distribution

The gamma distribution is most commonly used to model the time to the k^{th} event, when such an event is modeled by a Poisson process with rate parameter λ. Whereas the Poisson distribution is typically used to model the *number of events* in a period of given length, the gamma distribution models the *time to the k^{th} event* (or alternatively the time separating the k^{th} and $k^{th}+1$ events).



The gamma distribution is specified by the Poisson rate variable, λ , and the event number, k. The random variable, denoted as x, is the time period to the k^{th} event. Within GoldSim, the gamma distribution is specified by the mean and the standard deviation, which can be computed as a function of λ and k.

pdf: $$f(x) = \frac{\lambda(\lambda x)^{k-1} e^{-\lambda x}}{\Gamma(k)}$$

cdf: $$F(x) = \frac{\Gamma(k, \lambda x)}{\Gamma(k)}$$

where: $$\Gamma(k) = \int_0^\infty e^{-u} u^{k-1} du$$ (gamma function)

$$\Gamma(k, x) = \int_0^x e^{-u} u^{k-1} du$$ (incomplete gamma function)

$$k = \frac{\mu^2}{\sigma^2}$$

$$\lambda = \frac{\mu}{\sigma^2}$$

mean: μ

variance: σ^2

k and λ are also sometimes called α and β, respectively, and referred to as the shape and scale parameters.

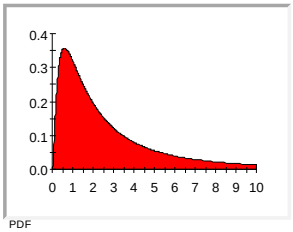
Note that

mean: α/β

variance: α/β^2

If α is near zero, the distribution is highly skewed. For α=1, the gamma distribution reduces to an exponential(β^{-1}) distribution. If α=n/2 and β = ½, the distribution is known as a chi-squared distribution with n degrees of freedom.

Log-Normal Distribution



The log-normal distribution is used when the *logarithm* of the random variable is described by a normal distribution. The log-normal distribution is often used to describe environmental variables that must be positive and are positively skewed.

In GoldSim, the log-normal distribution may be based on either the true mean and standard deviation, or on the geometric mean (identical to the median) and the geometric standard deviation. Thus, if the variable x is distributed log-normally, the mean and standard deviation of log x may be used to characterize the log-normal distribution. (Note that either *base 10* or *base e* logarithms may be used).

pdf: $$f(x) = \frac{1}{\zeta x \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\ln(x)-\lambda}{\zeta}\right)^2}$$

where:

$$\zeta^2 = \ln\left[1+\left(\frac{\sigma}{\mu}\right)^2\right]$$ (variance of ln x);

ζ is referred to as the shape factor; and

$$\lambda = \ln(\mu) - \frac{1}{2}\zeta^2$$ (expected value of ln x)

cdf: No closed form solution

mean (arithmetic): $$\mu = \exp\left[\lambda + \frac{1}{2}\zeta^2\right]$$

The mean computed by the above formula is the expected value of the log-normally distributed variable x and is a function of the mean and standard deviation of lnx. The mean value can be estimated by the arithmetic mean of a sample data set.

variance (arithmetic): $$\sigma^2 = \mu^2\left[\exp(\zeta^2) - 1\right]$$

The variance computed by the above formula is the variance of the log-normally distributed variable x. It is a function of the mean of x and the standard deviation of lnx. The variance of x can be estimated by the sample variance computed arithmetically.

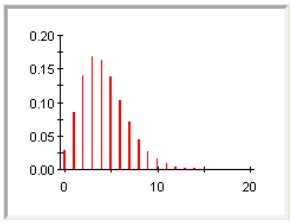
Other useful formulas:

Geometric mean = e^{λ}

Geometric standard deviation = e^{ζ}

A commonly used descriptor for a log-normal distribution is its Error Factor (EF), where the EF is defined as (geometric standard deviation) ^ 1.645. 90% of the distribution lies between Median/EF and Median*EF.

Negative Binomial Distribution



The negative binomial distribution is a discrete distribution specified by a number of successes (n), which can be fractional, and a probability of success (p). This distribution can be used to model the number of failures that occur when trying to achieve a given number of successes, and is used frequently in actuarial models.

pmf: $P(x)=\binom{x+n-1}{x} p^n (1-p)^x$ x = 0, 1, 2, 3...

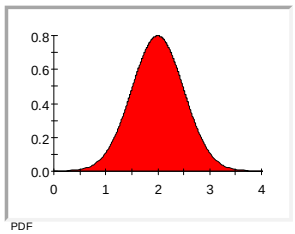
where: $\binom{x+n-1}{x} = \frac{(x+n-1)!}{x!(n-1)!}$

cdf: $F(x)= p^n \sum_{i=0}^{x} \binom{i+n-1}{i} (1-p)^i$

mean: $\frac{n(1-p)}{p}$

variance: $\frac{n(1-p)}{p^2}$

Normal Distribution



The normal distribution is specified by a mean (μ) and a standard deviation (σ). The linear normal distribution is a bell shaped curve centered about the mean value with a half-width of about four standard deviations. Error or uncertainty that can be higher or lower than the mean with equal probability may be satisfactorily represented with a normal distribution. The uncertainty of average values, such as a mean value, is often well represented by a normal distribution, and this relation is further supported by the *Central Limit Theorem* for large sample sizes.

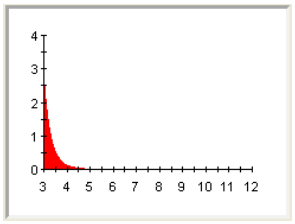
pdf: $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

cdf: No closed form solution

mean: μ

variance: σ^2

Pareto Distribution



The Pareto distribution is a continuous, long-tailed distribution specified by a shape parameter (a) and a scale parameter (b). The shape parameter and the scale parameter must be greater than zero. This distribution can be used to model things like network traffic in a telecommunications system or income levels in a particular country.

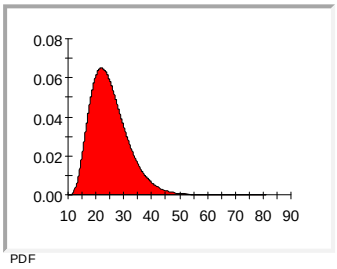
pdf: $$f(x)=\begin{cases}\dfrac{ab^a}{x^{a+1}} & if \quad x \geq b \\ 0 & otherwise\end{cases}$$

cdf: $$F(x)=\begin{cases}1-\left(\dfrac{b}{x}\right)^a & if \quad x \geq b \\ 0 & otherwise\end{cases}$$

mean: $$\frac{ab}{a-1}$$

variance: $$\frac{ab^2}{(a-1)^2(a-2)}$$

Pearson Type III Distribution



Often used in financial and environmental modeling, the Pearson Type III distribution is a continuous distribution specified by location (α), scale (β) and shape (p) parameters. Both the scale and shape parameters must be positive.

Note that the Pearson Type III distribution is equivalent to a gamma distribution if the location parameter is set to zero.

pdf: $$f(x)= \frac{1}{\beta\Gamma(p)}\left(\frac{x-\alpha}{\beta}\right)^{p-1} e^{-\left(\frac{x-\alpha}{\beta}\right)} \quad \text{if } x \geq \alpha$$

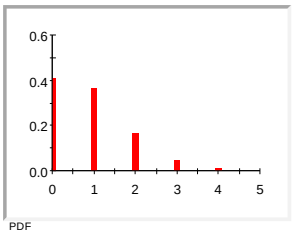
$$0 \quad \text{if } x < \alpha$$

cdf: $$F(x) = \frac{\Gamma\left(p, \frac{x-a}{\beta}\right)}{\Gamma(p)}$$

mean: $$\mu = \alpha + p\beta$$

variance: $$\sigma^2 = p\beta^2$$

Poisson Distribution



The Poisson distribution is a discrete distribution specified by a mean value, μ. The Poisson distribution is most often used to determine the probability for one or more events occurring in a given period of time. In this type of application, the mean is equal to the product of a rate parameter, λ, and a period of time, ω. For example, the Poisson distribution could be used to estimate probabilities for numbers of earthquakes occurring in a 100 year period. A rate parameter characterizing the number of earthquakes per year would be needed for input to the distribution. The time period would simply be equal to 100 years.

pdf: $f(x)=\frac{e^{-\mu}\mu^{x}}{x!}$ x = 0, 1, 2, 3...

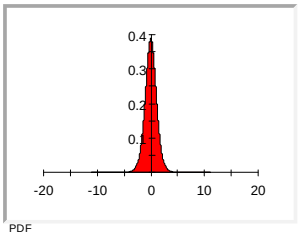
cdf: $F(x) = e^{-\mu}\sum_{i=0}^{x}\frac{\mu^{i}}{i!}$

mean: $\mu = \lambda\omega$

variance: $\sigma^2 = \mu$

where λ and ω are the "rate" and "time period" parameters, respectively. Note that quotations are used because the terminology rate and time period applies to only one application of the Poisson distribution.

Student's t Distribution



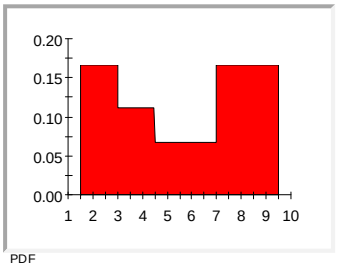
The Student's t distribution requires a single input: the number of degrees of freedom, which equals the number of samples minus one.

mean: 0

variance: $\frac{\nu}{\nu-2}$

where ν is the number of degrees of freedom

Sampled Result Distribution

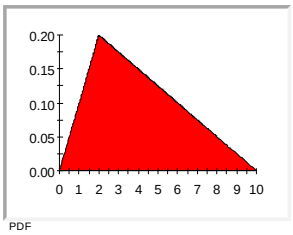


The sampled result distribution allows you to construct a distribution using observed results. GoldSim generates a CDF by sorting the observations and assuming that a cumulative probability of 1/(Number of Observations) exists between each data point. If there are multiple data points at the same value, a discrete probability equal to (N)/(Number of Observations) is applied at the value, where N is equal to the number of identical observations.

If the Extrapolation option is cleared, a discrete probability of 0.5/(Number of observations) is assigned to the minimum and maximum values. When the extrapolation option is selected, GoldSim extends the generated CDF to cumulative probability levels of 0 and 1 using the slope between the two smallest and two largest unique observations.

Triangular Distribution

The triangular distribution is specified by a minimum value (a), a most likely value (b), and a maximum value (c).



pdf:
$$f(x) = \begin{cases} \dfrac{2(x-a)}{(b-a)(c-a)} & a \le x \le b \\ \dfrac{2(c-x)}{(c-b)(c-a)} & b \le x \le c \\ 0 & x < a \text{ or } x > c \end{cases}$$

cdf:
$$F(x) = \begin{cases} 0 & x < a \\ \dfrac{(x-a)^2}{(b-a)(c-a)} & a \le x \le b \\ 1 - \dfrac{(c-x)^2}{(c-b)(c-a)} & b < x < c \\ 1 & x \ge c \end{cases}$$

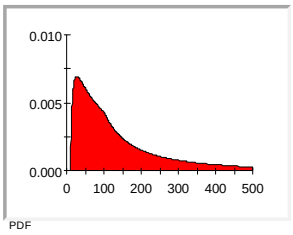
mean:
$$\mu = \frac{a+b+c}{3}$$

variance:
$$\sigma^2 = \frac{a^2+b^2+c^2-ab-ac-bc}{18}$$

Note that if the triangular is defined using the 10th and 90th percentile (instead of a minimum and a maximum) the minimum and maximum are estimated through iteration.

Log-Triangular Distribution

The log-triangular distribution is used when the *logarithm* of the random variable is described by a triangular distribution. The minimum (a), most likely (b), and maximum (c) values are specified in linear space.



pdf:
$$f(x) = \begin{cases} \dfrac{2\left[\ln\left(\frac{x}{a}\right)\right]\left(\frac{1}{x}\right)}{\ln\left(\frac{b}{a}\right)\ln\left(\frac{c}{a}\right)} & a \le x \le b \\ \dfrac{2\left[\ln\left(\frac{c}{x}\right)\right]\left(\frac{1}{x}\right)}{\ln\left(\frac{c}{a}\right)\ln\left(\frac{c}{b}\right)} & b \le x \le c \\ 0 & \text{otherwise} \end{cases}$$

cdf:
$$F(x) = \begin{cases} 0 & x<a \\ \dfrac{\left[\ln\left(\frac{x}{a}\right)\right]^2}{\ln\left(\frac{b}{a}\right)\ln\left(\frac{c}{a}\right)} & a \le x \le b \end{cases}$$

$$1 - \frac{\left[\ln\left(\frac{c}{x}\right)\right]^2}{\ln\left(\frac{c}{a}\right)\ln\left(\frac{c}{b}\right)} \qquad b < x \le c$$

$$1 \qquad x > c$$

mean: $$\mu = \frac{2}{d_1}\left\{a + b\left[\ln\left(\frac{b}{a}\right) - 1\right]\right\} + \frac{2}{d_2}\left\{c + b\left[\ln\left(\frac{b}{c}\right) - 1\right]\right\}$$

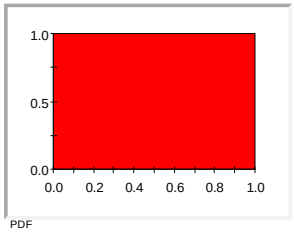
variance: $$\sigma^2 = \frac{2}{d_1}\left\{\frac{a^2}{4} + \frac{b^2}{2}\left[\ln\left(\frac{b}{a}\right) - \frac{1}{2}\right]\right\} + \frac{2}{d_2}\left\{\frac{c^2}{4} + \frac{b^2}{2}\left[\ln\left(\frac{b}{c}\right) - \frac{1}{2}\right]\right\} - \mu^2$$

where: $$d_1 = \ln\left(\frac{c}{a}\right)\ln\left(\frac{b}{a}\right) \text{ and}$$

$$d_2 = \ln\left(\frac{c}{a}\right)\ln\left(\frac{c}{b}\right)$$

Note that if the log-triangular is defined using the 10th and 90th percentile (instead of a minimum and a maximum) the minimum and maximum are estimated through iteration.

Uniform Distribution



The uniform distribution is specified by a minimum value (a) and a maximum value (b). Each interval between the endpoints has equal probability of occurrence. This distribution is used when a quantity varies uniformly between two values, or when only the endpoints of a quantity are known.

pdf: $$f(x) = \frac{1}{b-a} \qquad a \le x \le b$$

$$0 \qquad \text{otherwise}$$

cdf: $$F(x) = 0 \qquad x < a$$

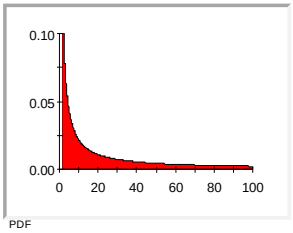
$$\frac{x-a}{b-a} \qquad a \le x \le b$$

$$1 \qquad x > b$$

mean: $$\mu = \frac{b+a}{2}$$

variance: $$\sigma^2 = \frac{(b-a)^2}{12}$$

Log-Uniform Distribution



The log-uniform distribution is used when the *logarithm* of the random variable is described by a uniform distribution. Log-uniform is the distribution of choice for many environmental parameters that may range in value over two or more log-cycles and for which only a minimum value and a maximum value can be reasonably estimated. The log-uniform distribution has the effect of assigning equal probability to the occurrence of intervals within each of the log-cycles. In contrast, if a linear uniform distribution were used, only the intervals in the upper log-cycle would be represented uniformly.

pdf: $f(x) = \dfrac{1}{x(\ln b - \ln a)}$ $\quad a \le x \le b$

0 $\quad x \le a$ or $x \ge b$

cdf: $F(x) = 0$ $\quad x \le a$

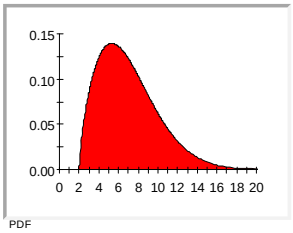
$\dfrac{\ln x - \ln a}{\ln b - \ln a}$ $\quad a \le x \le b$

1 $\quad x > b$

mean: $\mu = \dfrac{b-a}{(\ln b - \ln a)}$

variance: $\sigma^2 = \dfrac{b^2 - a^2}{2(\ln b - \ln a)} - \left(\dfrac{b-a}{(\ln b - \ln a)}\right)^2$

Weibull Distribution



The Weibull distribution is typically specified by a minimum value (ε), a scale parameter (β), and a slope or shape parameter (α). The random variable must be greater than 0 and also greater than the minimum value, ε.

The Weibull distribution is often used to characterize failure times in reliability models. However, it can be used to model many other environmental parameters that must be positive. There are a variety of distribution forms that can be developed using different values of the distribution parameters.

pdf: $f(x) = \dfrac{\alpha}{\beta - \varepsilon}\left(\dfrac{x - \varepsilon}{\beta - \varepsilon}\right)^{\alpha-1} e^{-\left(\frac{x-\varepsilon}{\beta-\varepsilon}\right)^{\alpha}}$

cdf: $F(x) = 1 - e^{-\left(\frac{x-\varepsilon}{\beta-\varepsilon}\right)^{\alpha}}$

mean: $\mu = \varepsilon + (\beta - \varepsilon)\Gamma\left(1 + \dfrac{1}{\alpha}\right)$

variance: $\sigma^2 = (\beta - \varepsilon)^2\left[\Gamma\left(1 + \dfrac{2}{\alpha}\right) - \Gamma^2\left(1 + \dfrac{1}{\alpha}\right)\right]$

The Weibull distribution is sometimes specified using a *shape parameter*, which is simply β - ε. Within GoldSim, the Weibull is defined by ε, α, and the mean-ε. As shown above, the mean can be readily computed as a function of ε, α, and β.

In practice, the Weibull distribution parameters are moderately difficult to determine from sample data. The easiest approach utilizes the cdf, fitting a regression through the sample data to estimate α (regression slope) and the difference quantity, $\beta - \varepsilon$.

Representing Truncated Distributions

Several distributions in GoldSim can be truncated at the ends (normal, log-normal, Gamma, and Weibull). That is, by specifying a lower bound and/or an upper bound, you can restrict the sampled values to lie within a portion of the full distribution's range.

The manner in which truncated distributions are sampled is straightforward. Because each point in a full distribution corresponds to a specific cumulative probability level between 0 and 1, it is possible to identify the cumulative probability levels of the truncation points. These then define a scaling function which allows sampled values to be mapped into the truncated range.

In particular, suppose the lower bound and upper bound were L and U, respectively. Any sampled random number R (representing a cumulative probability level between 0 and 1) would then be scaled as follows:

L + R(U-L)

The resulting "scaled" cumulative probability level would then be used to compute the sampled value for the distribution. The scaling operation ensures that it falls within the truncated range.

Correlation Algorithms

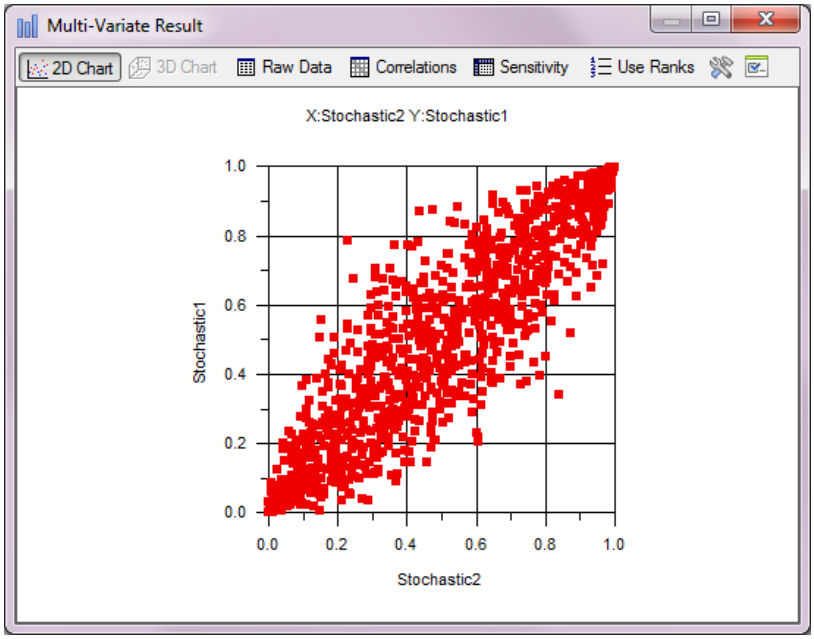
Several GoldSim elements that are used to represent uncertainty or stochastic behavior in models permit the user to define correlations between elements or amongst the items of a vector-type element.

To generate sampled values that reflect the specified correlations GoldSim uses copulas and the Iman and Conover methodology.

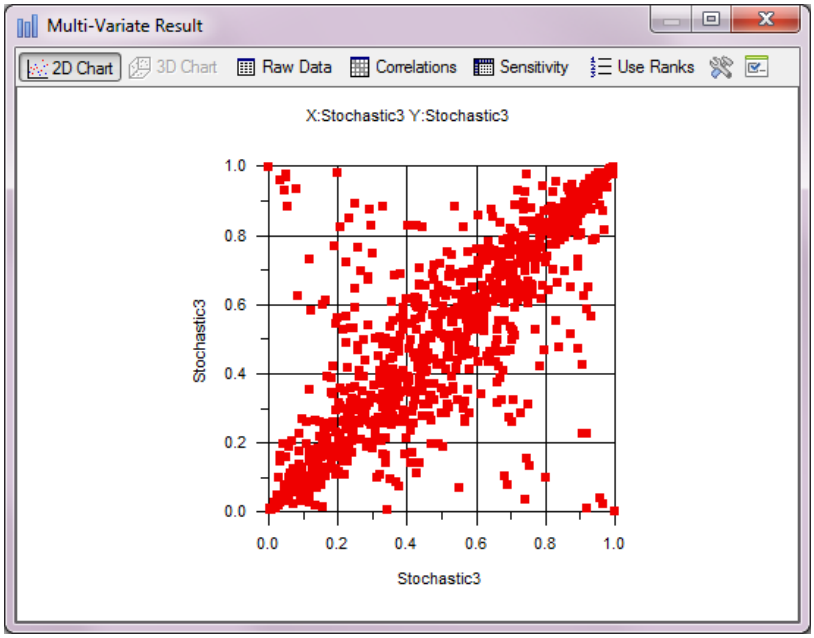
A copula is a function that joins two or more univariate distributions to form a multivariate distribution. As such, it provides a method for specifiying the correlation between two variables. Copulas are described in general terms by Dorey (2006) and in detail by Embrechts, Lindskog and McNeil (2001).

GoldSim uses two different copulas to generate correlated values: the Gaussian copula and the t-distribution copula. When a Stochastic element is correlated to itself, or to another Stochastic element, GoldSim uses the Gaussian copula to generate the correlated value. A vector-type Stochastic or History Generator can use either the Gaussian copula or a t-distribution copula to generate correlated values.

The Gaussian copula produces values where the correlation between variables is stronger towards the middle of the distributions than it is at the tails. The plot below shows the value at the first timestep of two variables (uniform distributions between 0 and 1) generated using the Gaussian copula with a correlation coefficient of 0.9:



A t-distribution copula produces a correlation that is stronger at the tails than in the middle. The plot below shows the value at the first timestep of two variables generated using the t-distribution copula for the same variables with a correlation coefficient of 0.9 and the **Degrees of Freedom** setting in the copula of 1):



The t-distribution's form is often what is observed in the real world: correlations at the extremes (e.g. representing a rare, but significant event such as a war) tend to be higher than correlations in the middle (representing the higher variability in more common occurrences).

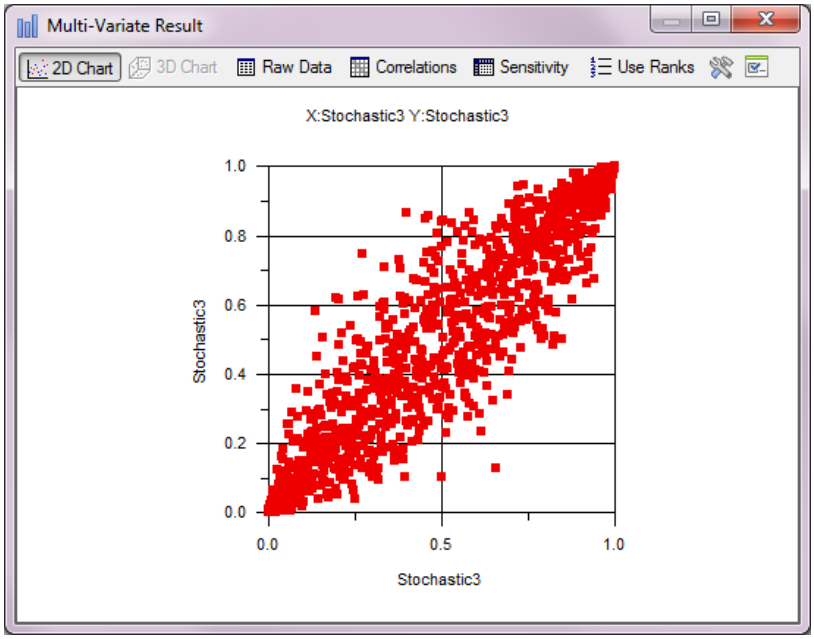
The **Degrees of Freedom** setting controls the tail dependency in the copula. A low value produces stronger dependence in the tails, while higher values produce stronger correlations in the middle of the distributions. This means that

a t-distribution copula with a high number of Degrees of Freedom will begin to behave like a Gaussian copula.

One of the weaknesses of the copula approach to generating correlated samples is that it does not respect Latin Hypercube Sampling (with the exception of the first item in a vector-type stochastic where the Gaussian copula is used to generate sampled values).

The Iman and Conover approach is designed to produce a set of correlated items that each respect Latin Hypercube sampling. Complete details on the algorithm's methodology can be found in Iman and Conover (1982).

Its behavior is similar, but not identical, to a Gaussian for the first sample:



However, if the element is resampled during a realization, elements that use the Iman and Conover approach will use the Gaussian copula to generate the second and subsequent sets of sampled data.

Sampling Techniques

This section discusses the techniques used by GoldSim to sample elements with random behavior. These include the following GoldSim elements:

- Stochastic
- Random Choice
- Timed Event Generator
- Event Delay
- Discrete Change Delay
- History Generator
- Source (in the Radionuclide Transport Module)
- Action element (in the Reliability module)
- Function element (in the Reliability module)

After first discussing how GoldSim generates and assigns random numbers, two enhanced sampling techniques provided by GoldSim (Latin Hypercube sampling and importance sampling) are discussed.

Generating and Assigning Random Number Seeds

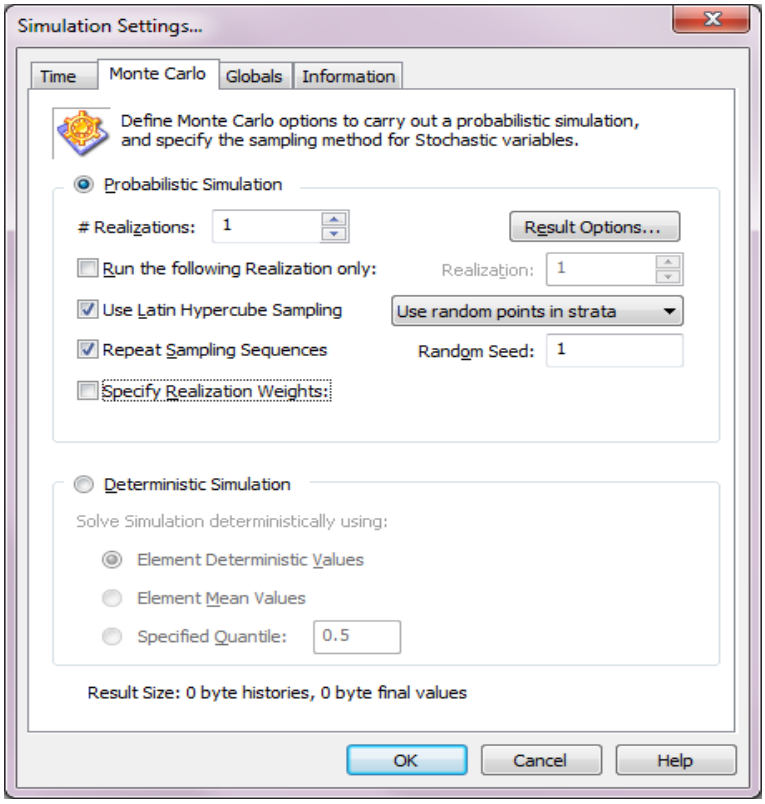
At the heart of GoldSim's random number generation are unique 'seeds' that are properties of:

- The model itself.
- Each element in the model

These seeds are assigned randomly when a new model or a new element is created, and are permanent properties of the model and the elements. However no two elements in a GoldSim model can have the same unique seed. This means that if an element is copied and pasted into a model where no elements have the same seed value, its seed will be unchanged. However, if it is pasted into the same model, or into a model where another element already has that seed value, one of the elements with the same seed value will be given a new unique seed. (Element seeds can be displayed by selecting the element in the graphics pane and pressing **Ctrl-Alt-Shift-F12**).

GoldSim's random number generation process is based on a method presented by L'Ecuyer (1988). Each seed actually consists of two long (32-bit) integers. When random numbers are generated, each of the integers cycles through a standard linear congruential sequence, with different constants used for the two sequences (so that their repeat-cycles are different). The random numbers that are generated are a function of the combination of the two seeds, so that the length of their repeat period is extremely long.

When a simulation is started, GoldSim generates a *simulation seed* which is the basis for all random numbers used in the simulation. If you have chosen to **Repeat Sampling Sequences** in the **Monte Carlo** tab of the Simulation Settings dialog, the simulation seed is the user-specified **Random Seed** specified in the dialog:



In this case, you can rerun a probabilistic simulation and get identical results. If you have chosen not to repeat sequences, the simulation seed is based on the system clock. In this case, each time you rerun the simulation, you will have a different simulation seed (and hence will generate slightly different results).

For each realization, GoldSim generates a *realization seed* based on the simulation seed. Each element in the model uses a combination of its element seed and this realization seed to start its random sequence for the realization. At the start of each Monte Carlo realization, the realization seed is incremented, and forms the basis for the element seeds to be used for that realization.

If the user selects the option to run a specified realization, such as realization 16, GoldSim simply iterates the realization seed the necessary number of times prior to starting the specified realization.

At the start of each realization, each element generates a new seed by combining its unique seed with the realization seed. This seed forms the basis for sampling of the element during the realization. Many elements have no random properties and hence are never sampled. Some elements, such as Stochastic elements, are typically sampled once, at the beginning of the realization. Other elements, such as Timed Events, Delay elements, and Stochastic elements that are specified to be resampled, are sampled multiple times.

Latin Hypercube Sampling

GoldSim provides an option to implement a Latin Hypercube sampling (LHS) scheme (in fact, it is the default when a new GoldSim file is created). The LHS option results in forced sampling from each "stratum" of each parameter.

The following elements use LHS sampling:

- Stochastic
- Random Choice

- Timed Event Generator
- Action (in the Reliability Module)
- Function (in the Reliability Module)

Each element's probability distribution (0 to 1) is divided into up to 10000 equally likely strata or slices (actually, the lesser of the number of realizations and 10000). The strata are then "shuffled" into a random sequence, and a random value is then picked from each stratum in turn. This approach ensures that a uniform spanning sampling is achieved.

Note: If possible, GoldSim will attempt to create LHS sequences where subsets are also complete LHS sequences. This means that if the total number of realizations is an even number, the first half and second half of the realizations are complete LHS sequences. If the total number of realizations is divisible by 4 or 8, that fraction of the total number of realizations, run in sequence, will be complete LHS sequences. This property of GoldSim's LHS sequences is sometimes useful for statistical purposes and also permits a user to extract a valid LHS sequence from a partially completed simulation by screening realizations. The details of this approach are discussed below ("Latin Hypercube Subsets").

Note that each element has an independent sequence of shuffled strata that are a function of the element's internal random number seed and the number of realizations in the simulation. If the number of realizations exceeds 10,000, then at the 10,001st realization each element makes a random jump to a new position in its strata sequence. A random jump is repeated every 10,000 realizations.

If you select "Use mid-points of strata" in the Simulation Settings dialog in models with less than 10,000 realizations, GoldSim will use the expected value of the strata selected for that realization. (Even if this option is selected, in simulations with greater than 10,000 realizations, the 10,001 and subsequent realizations will use random values from within the strata selected for that realization.) Using mid-points provides a slightly better representation of the distribution, but without the full randomness of the original definition of Latin Hypercube sampling (as described by McKay, Conover and Beckman, 1979).

If you select "Use random points in strata" in the Simulation Settings dialog, GoldSim also picks a random value from each stratum.

Note: LHS is only applied to the first sampled random value in each realization. Subsequent samples will not use LHS.

LHS appears to have a significant benefit only for problems involving a few independent stochastic parameters, and with moderate numbers of realizations. In no case does it perform worse than true random sampling, and accordingly LHS sampling is the default for GoldSim.

Note that the binary subdivision approach (described in more detail below) and the use of mid-stratum values are GoldSim-specific modifications to the original description of Latin Hypercube Sampling, as described in McKay, Conover and Beckman (1979).

Latin Hypercube Subsets

In order to allow users to do convergence tests, GoldSim's LHS sampling automatically organizes the LHS strata for each random variable so that binary

subsets of the overall number of realizations each represent an independent LHS sample over the full range of probabilities.

For example, if the user does 1,000 realizations, GoldSim will generate strata such that:

- Realizations 1-125 represent a full LHS sample with 125 strata. Realizations 126-250, 251-375, etc. through 876-1000 also represent full LHS samples with 125 strata each.
- Also, realizations 1-250, 251-500, 501-750, and 751-1000 represent full LHS samples with 250 strata each.
- And, realizations 1-500 and 501-1000 represent full LHS samples with 500 strata each.

The generation of binary subsets is automatic, and is carried out whenever the total number of realizations is an even number. Up to 16 binary subsets will be generated, if the number of realizations can be subdivided evenly four times. For example, if the total number of realizations was 100 then GoldSim would generate 2 subsets of 50 strata each and 4 subsets of 25 strata. If the total number of realizations was 400 then GoldSim would generate 2 subsets of 200 strata, 4 subsets of 100 strata, 8 subsets of 50 strata, and 16 subsets of 25 strata.

The primary purpose of this sampling approach is to use the subsets to carry out statistical tests of convergence. For example, the mean of each of the subsets of results could be evaluated and a t-test used to estimate statistics of the population mean, as described in Iman (1982). Rather than carrying out a set of independent LHS simulations using different random seeds, this approach allows the user to run a single larger simulation, with the benefits of a better overall representation of the system's result distribution, while still being able to test for convergence and to generate confidence bounds on the results. (A secondary benefit of the binary sampling approach is that if a simulation is terminated partway through it should have a slightly greater likelihood of having uniform sampling over the completed realizations than normal Latin Hypercube sampling would.)

The algorithm for assigning strata to the binary subsets is quite simple. For each pair of strata (e.g., 1st and 2nd; 3rd and 4th), the first member of the pair is randomly assigned to one of two "piles" ("left" or "right"), and the second member is assigned to the opposite pile. That is, conceptually, it can be imagined that the full set of strata is sent one at a time to a 'flipper' that randomly chooses 'left' or 'right' on its first, third, fifth etc. activations, and on its second, fourth, sixth etc. activations chooses the opposite of the previous value.

For the case of one binary subdivision of a total of N_s strata, the algorithm goes through the strata sequentially from lowest to highest, and passes them to a flipper that generates two 'piles' of strata. Each pile will therefore randomly contain one of the first two strata, one of the second two, and so on. Thus, each pile will contain one sample from each of the strata that would have been generated if only N_s /2 total samples were to be taken. The full sampling sequence is generated by randomly shuffling each pile and then concatenating the two sequences.

For four binary subdivisions the same approach is extended, with the first flipper passing its 'left' and 'right' outputs to two lower-level flippers. This results in four 'piles' of strata, which again are just randomly shuffled and then concatenated. The same approach is simply extended to generate eight or sixteen strata where possible.

Importance Sampling

For risk analyses, it is frequently necessary to evaluate the low-probability, high-consequence end of the distribution of the performance of the system. Because the models for such systems are often complex (and hence need significant computer time to simulate), and it can be difficult to use the conventional Monte Carlo approach to evaluate these low-probability, high-consequence outcomes, as this may require excessive numbers of realizations.

To facilitate these type of analyses, GoldSim allows you to utilize an ***importance sampling*** algorithm to modify the conventional Monte Carlo approach so that the high-consequence, low-probability outcomes are sampled with an enhanced frequency. During the analysis of the results which are generated, the biasing effects of the importance sampling are reversed. The result is high-resolution development of the high-consequence, low-probability "tails" of the consequences, without paying a high computational price.

The following elements permit importance sampling:

- Stochastic
- Random Choice
- Timed Event Generator
- Action (in the Reliability Module)
- Function (in the Reliability Module)

Note: Importance sampling is only applied to the first sampled random value in each realization for elements with importance sampling enabled. Subsequent samples will use random sampling.

Note: In addition to the importance sampling method described here (in which you can choose to force importance sampling on either the low end or high end of a Stochastic element's range), GoldSim also provides an advanced feature that supports custom importance sampling that can be applied over user-defined regions of the Stochastic element's range.

Read more: Customized Importance Sampling Using User-Defined Realization Weights (page 953).

Warning: Importance sampling affects the basic Monte Carlo mechanism, and it should be used with great care and only by expert users. In general, it is recommended that only one or at most a very few parameters should use importance sampling, and these should be selected based on sensitivity analyses using normal Monte Carlo sampling. **Importance sampling should only be used for elements whose distribution tails will *not* be adequately sampled by the selected number of realizations.**

How Importance Sampling Works

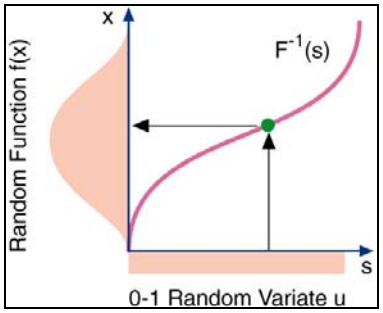
Importance sampling is a general approach to selectively enhance sampling of important outcomes for a model. In principle, the approach is simple:

1. Identify an important subset of the sampling space;
2. Sample that subset at an enhanced rate; and

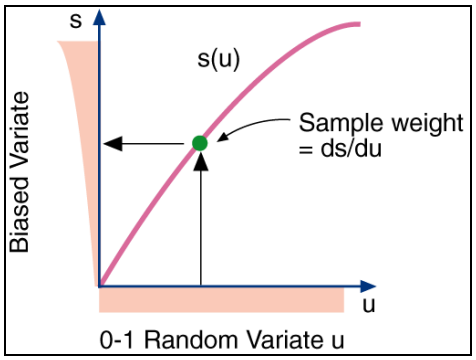
3. When analyzing results, assign each sample a weight inversely proportional to its enhancement-factor.

In conventional Monte Carlo sampling (with or without Latin Hypercube), each realization is assumed equally probable. It is straightforward, however, to incorporate a weight associated with each sample in order to represent the relative probability of the sample compared to the others.

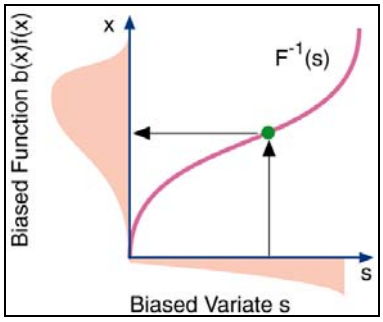
The conventional Monte Carlo approach is as shown below. A uniform 0 – 1 random variable **u** is sampled, and its value is then used as input to the inverse cumulative distribution function of the random variable:



In order to do importance sampling, the original uniformly-distributed random numbers are first mapped onto a non-uniform 'biased' sampling function **s**:



The biased variate **s** is then used to generate the random function. Since the input random numbers are no longer uniformly distributed, the resulting sample set is selectively enriched in high-consequence results:



In general, any continuous monotonic biasing function **s** which spans the range 0-1, and has **s**(0) = 0 and **s**(1) = 1 can be used to generate the set of input random numbers. The weight associated with each sampled realization is d**s**/d**u**, the slope of the biasing function **s** at the sampled point.

When a number of independent random variables are involved in a model, then the weight associated with a given realization is simply equal to the product of the weights of all parameters.

Biasing (Enhancement) Functions

GoldSim uses simple functions to selectively enhance either the upper or the lower end of an element's probability distribution.

The biasing function for enhancing the lower end of a distribution is:

$$s = u - \frac{u}{1+au} + \frac{u^2}{1+a}$$

where *a* is a function of the number of elements that are using importance sampling. This is equal to zero if only one element uses importance sampling, and is equal to ten times the number of importance sampled elements in all other cases. The effect of increasing *a* is to restrict the importance sampling to a smaller subset of the full range of the random value, which reduces the negative impacts of importance sampling numerous variables in the same model.

The sample weight is given by:

$$w = \frac{ds}{du} = 1 - \frac{1}{(1+au)^2} + \frac{2u}{1+a}$$

Note: For the first 10,000 realizations where GoldSim uses the expected values of the LHS strata the weight given to each sample is equal to the integral of s over the stratum divided by the corresponding integral of u over the strata. For the 10,001st and subsequent realizations GoldSim will calculate s and w for the sampled point.

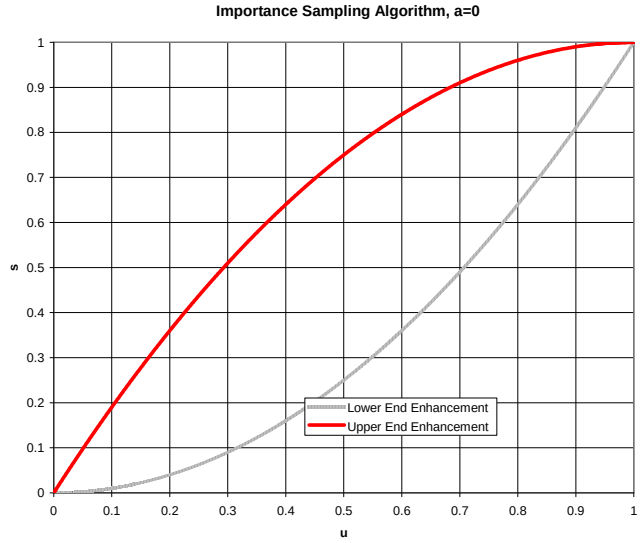
The biasing function for enhancing the upper end is:

$$s_{upper} = 1 - s_{lower}(1-u) = 1 - \left((1-u) - \frac{(1-u)}{1+a(1-u)} + \frac{(1+u)^2}{1+a} \right)$$

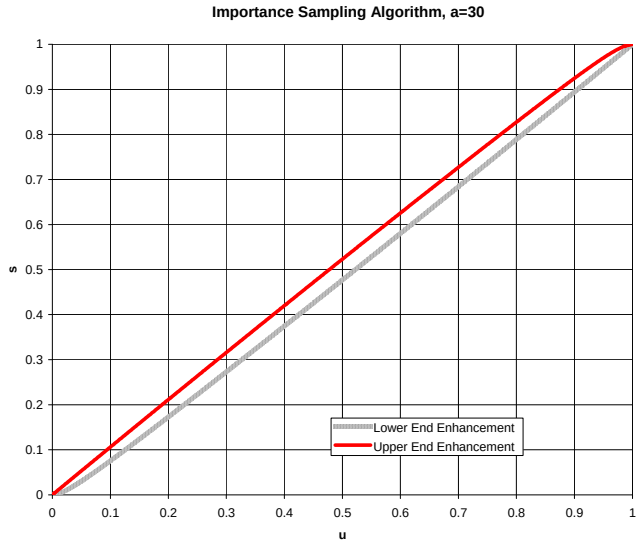
and the corresponding sample weight is given by:

$$w_{upper} = w_{lower}(1-u) = 1 - \frac{1}{(1+a(1-u))^2} + \frac{2(1-u)}{1+a}$$

The following plot shows the upper and lower biasing function when a single element utilizes importance sampling (an *a* value of zero):



The following figure shows the bias function when three elements are importance sampled (an *a* value of 30):



Note the less prominent bias as the number of importance sampled elements increases.

Behavior of Elements with Importance Sampling Enabled

The Stochastic element provides the option to choose between upper and lower end enhancement when importance sampling is enabled. However, the other elements that utilize importance sampling (the Timed Event element, the Reliability elements and the Random Choice element) importance sample only one end of the distribution.

Timed Events will use lower end importance sampling on the time to event distribution specified by the user. The Random Choice element has a slightly different behavior. When importance sampling is enabed, the Random Choice element sorts the probability of each outcome from lowest probability to highest probability and assigns them to sections of a uniform distribution. This uniform

distribution is then importance sampled at the lower end, so that the least probable outcomes are enhanced.

The Reliability elements will use a combination of these approaches. Time based failure modes behave in a similar manner to the timed event elements. Modes with a "probability of failure" perform importance sampling to enhance the number of failure outcomes.

It is important to note that only the first sampled value utilizes importance sampling. This means that only the first event from a Timed Event, the first Random Choice and the first occurrence of each Failure Mode will use importance sampling.

Representing Random (Poisson) Events

Timed Event elements can be specified to produce discrete event signals *regularly* or *randomly*.

Read more: Timed Event Elements (page 333).

Random events are simulated as occurring according to a Poisson process with a specified *rate of occurrence*. If an event occurs according to a Poisson process, the probability that N events will occur during a time interval T is described by the following expression (Cox and Miller, 1965):

$$P(N) = \frac{e^{-\lambda T}(\lambda T)^N}{N!}$$

where:

P(N) is the probability of N occurrences of the event within the time interval T;

T is the time interval of interest;

λ is the annual rate of occurrence; and

N is the number of occurrences of the event within the time interval T.

The expected (mean) number of occurrences during the time interval is equal to λT.

The Poisson distribution also has the property that the intervals between events are exponentially distributed (Benjamin and Cornell, 1970):

$$F(t) = 1 - e^{-\lambda t}$$

where F(t) is the probability that the time to the next event will be less than or equal to t.

If you indicate that the event can only occur once, GoldSim simulates the first occurrence according to the above equations, but does not allow the event to occur again.

Note that the rate of occurrence can be specified to be a function of time (i.e., the event process can be non-stationary).

Computing and Displaying Result Distributions

Displaying a CDF

Probabilistic results may be viewed in the form of a CDF (or a CCDF). This section describes how the values realized during the simulations are used to generate the CDF (or CCDF) seen by the user.

Creating the Results Array

Within GoldSim Monte Carlo results are stored in a particular data structure, referred to here as the *results array*. As the simulation progresses, each specific Monte Carlo realization result is added to the results array as a pair of values; the value realized and the weight given by GoldSim to the value. The array is filled "on the fly", as each new realization is generated. Theoretically, each separate realization would represent a separate entry in the results array (consisting of a value and a weight). If unbiased sampling were carried out each separate entry would have equal weight.

As implemented in GoldSim, however, the number of data pairs in the results array may be less than the number of realizations: There are two reasons why this may be the case:

- If multiple results have identical values, there is no need to have identical data pairs in the results array: the weight associated with the particular value is simply adjusted (e.g., if the value occurred twice, its weight would be doubled).
- For computational reasons, the results array has a maximum number of unique results which it can store. The maximum number for post-processing GoldSim simulation results is 25,000. If the number of realizations exceeds these limits, results are "merged" in a self-consistent manner. The process of merging results when the number of realizations exceeds 25,000 is discussed below.

To merge a new result with the existing results (in cases where the number of realizations exceeds one of the maxima specified above), GoldSim carries out the following operations:

- GoldSim finds the surrounding pair of existing results, and selects one of them to merge with. GoldSim selects this result based on the ratio of the distance to the result to the weight of the result (i.e., the program preferentially merges with closer, lower weight results).
- After selecting the result to merge with, GoldSim replaces its value with the weighted average of its existing value and the new value; it then replaces its weight with the sum of the existing and new weights.

There is one important exception to the merging algorithm discussed above: If the new result will be an extremum (i.e., a highest or a lowest), GoldSim replaces the existing extremum with the new one, and then merges the existing result instead. This means that GoldSim never merges data with an extremum.

Plotting a CDF

Plotting the CDF from the results array is straightforward. The basic algorithm assumes that the probability distribution between each adjacent pair of result values is uniform, with a total probability equal to half the sum of the weights of the values. One implication of this assumption is that for a continuous distribution the probability of being less than the smallest value is simply equal to half the weight of the lowest value and the probability of being greater than the highest value is half the weight of the highest value.

For example, if we have ten equally weighted results in a continuous distribution, there is a uniform probability, equal to 0.1, of being between any

two values. The probability of being below the lowest value or above the highest value would be 0.05. GoldSim extrapolates the value at a probability level of 0 using the slope between the first two observations. Similarly the slope between the last two observations is used to estimate the value at a probability level of 1.

Note: Extrapolation is not carried out for Milestone result distributions, Reliability element failure and repair time distributions, Decision Analysis results, and for Sampled Stochastic distributions if the option to extrapolate is not checked.

In certain circumstances there are several minor variations to the basic algorithm discussed above:

- If the number of distinct results is much smaller than the number of realizations, GoldSim assumes the distribution is discrete (rather than continuous), and lumps the probabilities at the actual values sampled. In particular, if the total number of unique results is <= 20, and more than 50% of the realization results were identical to an existing result, GoldSim presumes the distribution is discrete and plots it accordingly. The user can observe this by sampling from a binomial distribution.
- GoldSim uses a heuristic algorithm to decide if each specific result represents a discrete value: if the number of exact repetitions of a particular result exceeds 2(1 + average number of repetitions), GoldSim treats the result as a discrete value and does not 'smear' it. For example, suppose the result is 0.0 50% of the time, and normal (mean=10, s.d.=2) the rest of the time. The first result value would be 0.0, with a weight of about 0.5. The second value would be close to 8, with a weight of 1/# realizations. We would not want to smear half of the 0 result over the range from 0 to 8!

When the user selects the confidence bounds options (discussed below), a different algorithm is used to display and plot CDF values. In particular, the displayed value is simply the calculated median (50^{th} percentile) in the probability distribution for the "true" value of the desired quantile.

Displaying a PDF

Displaying PDFs is much more difficult than CDFs, as unless a large number of realizations are run, PDFs tend to be "noisy" even if the corresponding CDF appears smooth (small changes in the slope of the CDF are typically not noticeable to the eye, but these can translate into very noticeable "spikes" in the PDF).

To display a PDF, GoldSim creates a series of equally-spaced bins, with a constant density value within each bin. The density within each bin is the average slope of the CDF over the bin. The number of bins automatically increases with the number of realizations. In particular, the number of bins is equal to the square root of the number of results being considered. The maximum number of bins used is 1000 (and the minimum is 1).

Computing and Displaying Confidence Bounds on the Mean

GoldSim is able to perform a statistical analysis of Monte Carlo results to produce confidence bounds on the mean of a probabilistic result. These bounds reflect uncertainty in the probability distribution due to the finite number of Monte Carlo realizations - as the number of realizations is increased, the limits become narrower. The confidence bounds on the mean are displayed in the Statistics section of the Distribution Summary dialog when viewing Distribution Results if the **Confidence Bounds** checkbox is checked.

Read more: Viewing a Distribution Summary (page 600).

This approach to compute the confidence bounds uses the t distribution, which is strictly valid only if the underlying distribution is normal. The 5% and 95% confidence bounds on the population mean are calculated as defined below:

$$P\{\overline{X} + t_{0.05}\frac{s_x}{\sqrt{n}}\} < \mu = 0.05$$

and $$P\{\overline{X} + t_{0.95}\frac{s_x}{\sqrt{n}}\} \geq \mu = 0.05$$

where:

$\overline{X}$	is the sample mean
$t_{0.05}$	is the 5% value of the t distribution for n-1 degrees of freedom
$t_{0.95}$	is the 95% value, = $-t_{0.05}$.
s_x	is the sample standard deviation
μ	is the true mean of the population, and
n	is the number of samples (realizations).

As the number of realizations, n, becomes large, the Central Limit theorem becomes effective, the t distribution approaches the normal distribution, and the assumption of normality is no longer required. This may generally be assumed to occur for n in the order of 30 to 100 realizations even for results of highly-skewed distributions.

Computing and Displaying Confidence Bounds on CDFs and CCDFs

GoldSim is able to perform a statistical analysis of Monte Carlo results to produce confidence bounds on the resulting probability distribution curves. These bounds reflect uncertainty in the probability distribution due to the finite number of Monte Carlo realizations - as the number of realizations is increased, the limits become narrower. The confidence bounds are displayed when viewing Distribution Results if the **Show Confidence Bounds** button in the Distribution display window is pressed.

The confidence bounds appear as different-colored curves on the probability plots produced by the GoldSim user interface. The bounds represent 5% and 95% confidence limits on the distribution value at each probability level. Confidence bounds cannot be displayed for PDFs..

The theory used to produce the confidence bounds has several limitations:

- The bounds on distributions can only be calculated for cases where all realizations have equal weights. If importance sampling is used for any parameter is used, GoldSim will not display confidence bounds.
- The confidence bounds cannot be calculated for values less than the smallest result, or greater than the largest result. As a result of this, the confidence-bound curves do not generally reach all of the way to the tails of the result plots.

In cases with relatively few stochastic parameters, Latin Hypercube sampling can increase the accuracy of the probability distributions. The confidence bounds are not able to reflect this improvement, and as a result will be conservatively wide in such cases.

Theory: Bounds on Cumulative Probability

Suppose we have calculated and sorted in ascending order n random results r_i from a distribution. What can we say about the q^{th} quantile x_q (e.g., q=0.9) of the underlying distribution?

Each random result had a probability of q that its value would be less than or equal to the actual q^{th} quantile x_q. The total number of results less than x_q was therefore random and binomially distributed, with the likelihood of exactly i results <= x_q being:

$$P(i) = P(r_i \le x_q \le r_{i+1}) = \binom{n}{i} q^i (1-q)^{n-i} = \frac{n!}{i!(n-i)!} q^i (1-q)^{n-i}$$

Note that there may be a finite probability that the value of x_q is less than the first or greater than the largest result: for example, if 100 realizations r_i were sampled, there would be a probability of 0.366 that the 0.99 quantile exceeded the largest result. The 100 realization probability distribution for $x_{0.99}$ is as follows:

Between Results	Probability	Cumulative Probability
<94	0.0000	0.0000
94 and 95	0.0005	0.0005
95 and 96	0.003	0.0035
96 and 97	0.015	0.0185
97 and 98	0.061	0.0795
98 and 99	0.1849	0.2644
99 and 100	0.370	0.6344
>100	0.366	1.

GoldSim assumes that the probability defined by the equation presented above is uniformly distributed over the range from r_i to r_{i+1} , and interpolates into the Monte Carlo results-list to find the 5% and 95% cumulative probability levels for x_q. For example, for 100 realizations, the 5% confidence bound on the 0.9 quantile is 0.31 of the distance from result 85 to result 86, and the 95% confidence bound is 0.22 of the distance from result 95 to result 96.

Between Results	Probability	Cumulative Probability
<85	-	0.040
85 and 86	.033	0.073
86 and 87	0.051	0.124
87 and 88	0.074	0.198
88 and 89	0.099	0.297
89 and 90	0.120	0.417
90 and 91	0.132	0.549
91 and 92	0.130	0.679
92 and 93	0.115	0.794
93 and 94	0.089	0.883

Between Results	Probability	Cumulative Probability
94 and 95	0.060	0.942
95 and 96	0.034	0.976
96 and 97	0.016	0.992
97 and 98	0.006	0.998
98 and 99	0.0016	1.000
99 and 100	0.000	1.000
>100	0.000	1.000

Using the above probability distribution, it is also possible to calculate the expected value of $x_{q,}$. This approach appears (by experimentation) to provide a slightly more reliable estimate of x_q than the conventional Monte Carlo approach of directly interpolating into the results-list. The expected value of x_q is calculated by summing the product of the probability of x_q lying between each pair of results and the average of the corresponding pair of result-values, i.e.,

$$\bar{x}_q = \sum_{i=1}^{n} P(i)\frac{[r_{i+1} + r_i]}{2}$$

When using this equation to estimate a very high or low quantile, a problem arises when the probability level, q, is near to 0 or 1, as there can be a significant probability that x_q lies outside the range of results. In the first table presented above, for example, there is a 0.366 chance of $q_{0.99}$ exceeding the largest result. In such cases, an estimate of the expected value of x_q can be found by extrapolating from the probabilities within the range of the results. Obviously, however there are limits to extrapolation and without knowledge of the actual distributional form no extrapolation would produce a reliable estimate of (say) the 0.9999 quantile if only 100 realizations had been performed.

In evaluating the binomial distribution for large values of n, large numbers can be generated which can cause numerical difficulties. To avoid these difficulties, when the number of realizations (n) is greater than 100, GoldSim uses either the normal or Poisson approximations to the binomial distribution. The Poisson approximation is used when i or (n-i) is less than 20 and the normal distribution is used otherwise. These approximations are described in any introductory statistics text.

Computing the Conditional Tail Expectation

The Conditional Tail Expectation (CTE) is the expected value of the result given that it lies above a specified value or quantile (Hardy, 2006). The CTE is displayed in the 'Calculator' portions of the Stochastic and Result Distribution elements, and is an optional output type for a Monte Carlo SubModel element.

Read more: Specifying the Distribution for a Stochastic Element (page 158); Viewing a Distribution Summary (page 600); Creating the Output Interface to a SubModel (page 925).

For a tail starting at value k, the CTE is defined as:

$$CTE_k = \frac{1}{1 - F(k)} \int_k^{\infty} x \cdot f(x) \cdot dx$$

where:

F(k) is the cumulative probability of value k

f(x) is the probability density at value x

The CTE is related to another statistical measure of a distribution, the partial expectation (PE). The PE is defined as:

$$PE_k = \int_k^{\infty} (x-k) \cdot f(x) \cdot dx$$

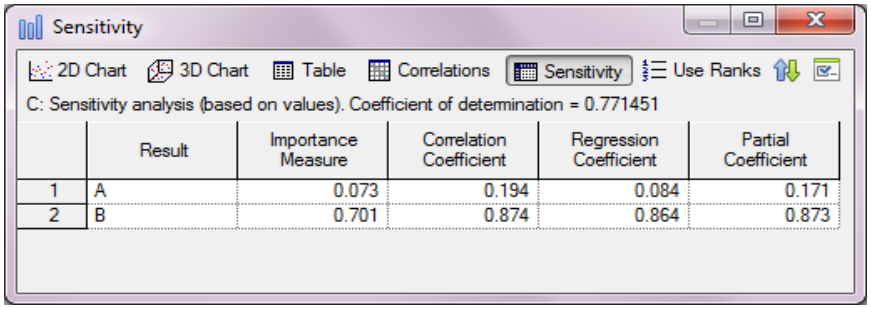
The CTE and PE are related as follows:

$$PE_k = (1 - F(k))[CTE_k - k]$$

$$CTE_k = \frac{PE_k}{(1 - F(k))} + k$$

Computing Sensitivity Analysis Measures

GoldSim provides a number of statistical sensitivity analyses through the multi-variate result display option. If you press the **Sensitivity Analysis…** button from the Multi-Variate Result dialog, a table such as this is displayed:



	Result	Importance Measure	Correlation Coefficient	Regression Coefficient	Partial Coefficient
1	A	0.073	0.194	0.084	0.171
2	B	0.701	0.874	0.864	0.873

This table displays measures of the sensitivity of the selected result (the output from which you selected **Multi-Variate Result…**) to the selected input variables.

Note: You can control whether the sensitivity analyses are based on the values or the ranks of the variables and the result via the **Use Ranks** button at the top of the display.

The measures that GoldSim computes are:

- Coefficient of determination;
- Correlation coefficient;
- Standardized regression coefficient (SRC);
- Partial correlation coefficient; and
- Importance measure.

The manner in which each of these measures is computed is described below.

Read more: Viewing a Sensitivity Analysis Table (page 639).

Computing the Coefficient of Determination

The coefficient of determination represents the fraction of the total variance in the result that can be explained based on a linear (regression) relationship to the input variables (i.e., Result = aX + bY + cZ + …). The closer this value is to 1, the better that the relationship between the result and the variables can be explained with a linear model.

The formulation for the coefficient of determination (R^2) can be found in Iman (1985). Note that if all of the variables on uncorrelated, then:

$$R^2 = \sum_i C_i$$

Where C_i is the correlation coefficient for variable i.

Computing Correlation Coefficients

Rank (Spearman) or value (Pearson) correlation coefficients range between -1 and +1, and express the extent to which there is a linear relationship between the selected result and an input variable.

The value correlation coefficient is computed as follows:

$$C_{rp} = \frac{\sum_{i=1}^{n}(p_i - m_p)(r_i - m_r)}{\sqrt{\sum_{i=1}^{n}(p_i - m_p)^2 \sum_{i=1}^{n}(r_i - m_r)^2}}$$

where:

C_{rp}= the value correlation coefficient;

n = the number of selected data points (realizations);

p_i = value of output p for realization i;

ri = value of output r for realization i;

m_p = mean value of output p; and

m_r = mean value of output r.

Note that the value correlation coefficient as defined here provides a measure of the *linear* relationship between two variables.

Furthermore, it may be affected by a few aberrant pairs (i.e., a good alignment of a few extreme pairs can dramatically improve an otherwise poor correlation coefficient; likewise, an otherwise good correlation could be ruined by the poor alignment of a few extreme pairs).

To overcome these problems, the value correlation coefficient can be supplemented by the rank correlation coefficient.

The rank correlation coefficient (also referred to as the *Spearman rank correlation coefficient*) is computed using the same equation as that of the value correlation coefficient with the *ranks* of the data values being used rather than the actual values:

$$C_{rp,rank} = \frac{\sum_{i=1}^{n}(Rp_i - m_{Rp})(Rr_i - m_{Rr})}{\sqrt{\sum_{i=1}^{n}(Rp_i - m_{Rp})^2 \sum_{i=1}^{n}(Rr_i - m_{Rr})^2}}$$

where:

$C_{rp,rank}$ = the rank correlation coefficient;

n = the number of selected data points (realizations);

Rp_i = the rank (from 1 to n) of output p for realization I;

Rr_i = the rank (from 1 to n)of output p for realization I;

m_{Rp} = mean value of the rank of output p; and

m_{Rr} = mean value of the rank of output r.

In GoldSim, the ranks of equal values are the same. For example, if the lowest two realizations of an output were the same, their ranks would both be 1.5 (the mean of 1 and 2), with the third lowest value being assigned a rank of 3.

Note: A correlation coefficient cannot be computed for a pair of outputs if one of the outputs has a standard deviation of zero (i.e., is constant). In this case, GoldSim sets the correlation coefficient to zero.

Computing Standardized Regression Coefficients (SRC)

Standardized regression coefficients range between -1 and +1and provide a normalized measure of the linear relationship between variables and the result. They are the regression coefficients found when all of the variables (and the result) are transformed and expressed in terms of the number of standard deviations away from their mean. GoldSim's formulation is based on Iman et al (1985).

The use of standardized regression coefficients to identify the importance of correlated input variables is described in Iman et al (1985) and Mishra (2004). In this approach the correlation matrix C for the input variables is augmented, with an additional row/column assigned for the result (GoldSim uses the first row/column for this purpose).

The standardized regression coefficients are based on the inverse of the augmented correlation matrix, and are found by dividing the terms in the augmented column of the matrix by the negative of the augmented diagonal term:

$$SRC_{y,i} = \frac{-c_{iy}}{c_{yy}}$$

where:

$SRC_{y,i}$ = the standardized regression coefficient of the result (Y) with respect to input variable X_i;

c_{iy} = the off-diagonal term in the inverted correlation matrix for row i, column y; and

c_{yy} = the diagonal term in the inverted correlation matrix corresponding to the result y.

Computing Partial Correlation Coefficients

Partial correlation coefficients reflect the extent to which there is a linear relationship between the selected result and an input variable, after removing the effects of any linear relationships between the other input variables and both the result and the input variable in question. For systems where some of the input variables may be correlated, the partial correlation coefficients represent the "unique" contribution of each input to the result. GoldSim's formulation is based on Iman et al (1985).

The partial correlation coefficient $P_{y,i}$ is calculated as:

$$P_{y,i} = \frac{-c_{iy}}{\sqrt{c_{ii}c_{yy}}}$$

where:

$P_{y,I}$ = the partial correlation coefficient of the result (Y) to input variable X_i;

c_{iy} = the off-diagonal term in the inverted correlation matrix for row i, column y; and

c_{ii}, c_{yy} = the diagonal terms in the inverted correlation matrix corresponding to the input variable and the result, respectively.

Note that if any two or more of the input variables are linearly dependent, for example if they are perfectly correlated, then the correlation matrix becomes singular and cannot be inverted. GoldSim, which uses Choleski decomposition to compute the inverse, will automatically adjust the correlation matrix if necessary in order to compute the partial correlation coefficients. This adjustment, which takes the form of 'weakening' of the off-diagonal terms, does not affect the displayed correlation coefficients.

Computing Importance Measures

If there is a nonlinear, non-monotonic relationship between an input variable and the result, conventional correlation coefficients may not reveal the relationship. The Importance Measure computed by GoldSim is a normalized version of the measure $E[V(Y|X_i)]$ discussed in Saltelli and Tarantola (2002). The Saltelli and Tarantola measure represents the expected value of the variance if the input variable X_i was not uncertain. Thus, the smaller this value, the more the input variable controls the result.

The GoldSim version of this measure is normalized as follows:

$$M_{y,i} = 1 - \frac{E\left[V_y\left(Y \mid X_i\right)\right]}{V_y}$$

where:

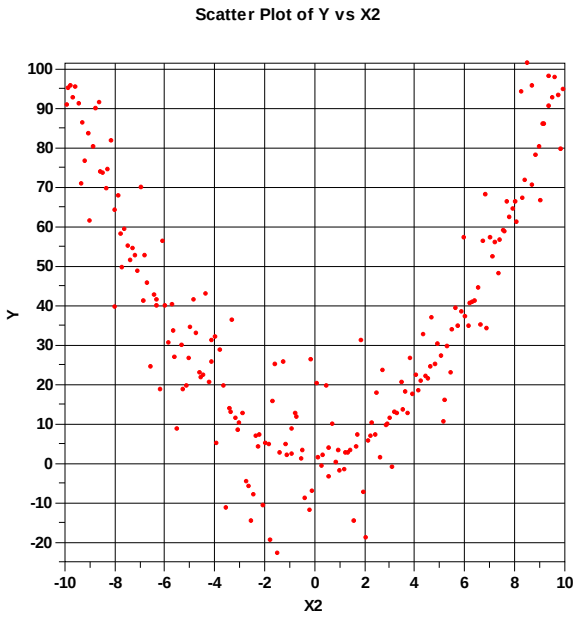
M_y = the GoldSim importance measure for the sensitivity of the result (Y) to input variable X_i;

V_y = the current variance in the result Y; and

$E[V_y (Y|X_i)]$ = the expected value of V_y if the input variable X_i was perfectly known.

Thus, GoldSim's Importance Measure $M_{y,i}$ represents the fraction of the result variance that is explained by X_i. Note that, like the correlation coefficients and scatter-plots, the importance measure can be calculated using either values or ranks, as specified in the main property window for the multivariate element.

GoldSim calculates $M_{y,i}$ numerically, using the following method. Construct a 2-d scatter plot of the results, with the X_i values on the horizontal axis and the result on the vertical axis. If X_i is very important to the result, then this plot will show a clustering around a central line or curve:



Subdivide the plot into N_s vertical segments based on the ranks of the X_i-values, where N_s equals the square root of the number of model realizations. For each of the segments, estimate the variance of Y within that segment by using a weighting function that assigns decreasing weights to the results as their distance from the center of the segment increases. Then compute the average value of the variance over all of the segments, i.e. $E[V_y (Y|X_i)]$. For the weighting function, GoldSim uses the density of a beta distribution having a mean equal to the X-value at the center of the segment, a standard deviation equal to the segment width, and upper and lower bounds corresponding to the range of X-values.

The example plot above is based on calculating:

$$Y = X_1 + X_2^{\,2} + X_3^{\,3}$$

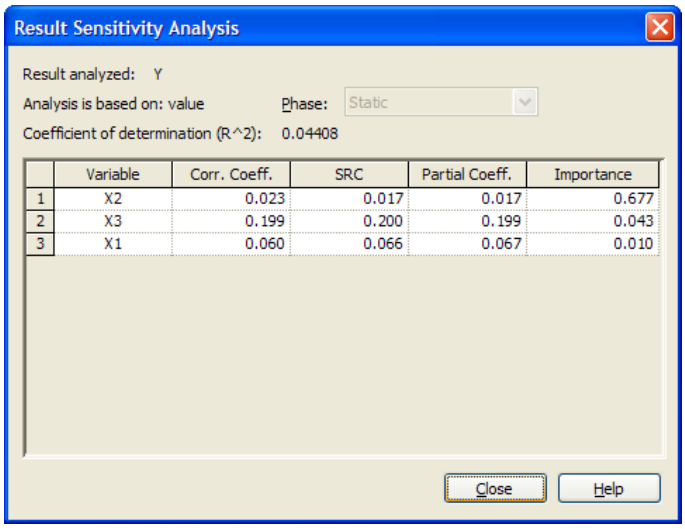
where:

X_1 = a random variable with a U(1,2) distribution;

X_2 = a random variable with a U(-10,10) distribution; and

X_3 = a random variable with a U(-3,3) distribution.

The plot shown above, plotting Y vs X_2, clearly reveals the importance of X_2 in this model. Conventional correlation analysis is completely unable to recognize this sensitivity, as indicated by the correlation coefficient of 0.023 in the screen-shot below. However, the importance measure is sensitive to it, and identifies X_2 as the most importance variable:



References

The references cited in this appendix are listed below.

Cox, D.R. and H.D. Miller, 1965. The Theory of Stochastic Processes, Chapman and Hall, New York.

Benjamin, J.R. and C.A. Cornell, 1970. Probability, Statistics, and Decision for Civil Engineers, McGraw-Hill, New York.

Hardy, Mary, 2006. *Simulating VaR and CTE*, Financial Engineering News, http://www.fenews.com/fen47/topics_act_analysis/topics-act-analysis.htm.

Iman, R.L. 1982. *Statistical Methods for Including Uncertainties Associated with the Geologic Isolation of Radioactive Waste Which Allow for a Comparison with Licensing Criteria*. Proceedings of the Symposium on Uncertainties Associated with the Regulation of the Geologic Disposal of High-Level Radioactive Waste, Gatlinburg, Tennessee, March 9-13, 1981. Kocher, D.C., ed. NUREG/CP-0022. 145-157. Washington, D.C.: U.S. Nuclear Regulatory Commission. TIC: 213069.

Iman, R.L. and W.J. Conover, 1982. *A Distribution-Free Approach to Inducing Rank Correlation Among Input Variables*, Communications in Statistics: Simulation and Computation, 11(3), pp 311-334,

Iman, R.L. et al., 1985. *A FORTRAN Program and User's Guide for the Calculation of Partial Correlation and Standardized Regression Coefficients*, NUREG/CR-4122, SAND85-0044.

L'Ecuyer, P., 1988, *Communications of the ACM*, vol. 31, pp. 742-744.

McKay, M.D., Conover, W. J., and Beckman, R. J., 1979. *A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code*, Technometrics, 21, 239-245.

Mishra, Srikanta, 2004. *Sensitivity Analysis with Correlated Inputs – an Environmental Risk Assessment Example*. Proceedings of the 2004 Crystal Ball User Conference, http://www.decisioneering.com/cbuc/2004/papers/CBUC04-Mishra.pdf.

Saltelli, A. and S. Tarantola, 2002. *On the Relative Importance of Input Factors in Mathematical Models: Safety Assessment for Nuclear Waste Disposal*, J. Am. Stat. Ass., Vol. 97, No. 459.

Appendix C: Implementing External (DLL) Elements

Begin at the beginning and go on till you come to the end; then stop.

Lewis Carroll, Alice in Wonderland

Appendix Overview

GoldSim allows you to develop separate program modules (written in C, C++, Pascal, FORTRAN or other compatible programming languages) which can then be directly coupled with the main GoldSim algorithms. These user-defined modules are referred to as *external functions*, and are linked into GoldSim as DLLs (Dynamic Link Libraries) at run time. GoldSim interfaces with the DLL via an *External element*.

Read more: External (DLL) Elements (page 873).

Integrating your external program module into GoldSim requires that you develop a "wrapper" or "shell" around the function and compile it into a DLL. This appendix discusses the details of how external functions must be coded and compiled.

In this Appendix

This appendix discusses the following:

- Understanding External (DLL) Elements
- Implementing an External Function
- External Function Examples
- External Function Calling Sequence
- DLL Calling Details

Understanding External (DLL) Elements

External functions work with External elements to do calculations or other manipulations that are not included in the standard capabilities of GoldSim. The external function facility allows special purpose calculations or manipulations to be accomplished with more flexibility, speed or complexity than with the standard GoldSim element types.

The external functions are bound to the GoldSim executable code at run time using DLL technology. The DLL files should be present in the same folder as the GoldSim .gsm file, in the same folder as the GoldSim executable file, or elsewhere in the user's path.

Note that these functions are external to GoldSim and are not covered by the standard GoldSim verification process. The user is responsible for any necessary testing of external functions.

Every external function is called by GoldSim with specific requests. The requests include initialization, returning the function version number, performing a normal calculation, and "cleaning up" after a simulation. The function name and argument list (the set of input and output data for the function) are specified by the GoldSim user when setting up the External element.

External functions should provide their own error handling, message handling, file management and memory management if required. It is essential that when it receives a "clean up" request, an external function should release any dynamically acquired memory and close any open files.

Note: In the case of an error condition, the external function should always return an error code to GoldSim, so that the user can be informed about the problem and the simulation can be terminated cleanly with no memory leaks.

Implementing an External Function

Important Restrictions

The implementation of the external function is left to the programmer, but several restrictions apply so that the functions can be correctly called by GoldSim. They are:

- The function return value is ignored. For example, use *void* functions in C/C++, or subroutines in FORTRAN.
- Data are passed from GoldSim to the external function and back again to GoldSim via arrays of double precision floating point input and output arguments.
- Input arguments ***must not*** be modified by the external function. Doing so may cause memory corruption in the DLL and/or GoldSim.
- Each function must manage its own initialization and memory allocation, and its own messages to the user (if any).
- Unique external function (or subroutine) names must be defined in each DLL. In C++, function names are case-sensitive, while in FORTRAN, the case of the external subroutine name is determined from the DLL

build options. In all instances, the function name specified in GoldSim is case-sensitive, and must agree with the case specified in the DLL.

- All files required to run your specific DLL must be properly installed on the machine where GoldSim is running. This includes any additional runtime libraries required by your DLL.
- Most development environments allow both ***static*** and ***dynamic*** binding of runtime libraries to DLLs. DLLs built with static binding are stand-alone, with all run-time library references included in the DLL file. However, if a DLL is built with dynamic binding, the runtime libraries are ***not included***. For a DLL with dynamic binding, the runtime libraries (e. g. **msvcrt.dll** for Visual C++ or **libifcoremd.dll** for Intel Visual Fortran) must be present and in the environment PATH variable on the machine where GoldSim is running.

External Function Format

When calling methods from the DLL, GoldSim always expects the following C/C++ function signature:

```
extern "C" void __declspec(dllexport)
MyExternalFcn(int     XFMethod,
int*    XFState,
double* inargs,
double* outargs)
```

The extern "C" specifies C language style linkage between GoldSim and the DLL, and __declspec(dllexport) makes the function visible outside the DLL.

For Intel Visual Fortran, the following subroutine signature can be used:

```
subroutine my_external_fcn(xfmethod, xfstate, inargs,
outargs)
    !DEC$ ATTRIBUTES dllexport,c :: add_mult_scalars
    !DEC$ ATTRIBUTES value       :: nmethod
    !DEC$ ATTRIBUTES reference   :: nstatus
    !DEC$ ATTRIBUTES reference   :: dinputs
    !DEC$ ATTRIBUTES reference   :: doutputs
```

Other FORTRAN development environments may require different attributes for the proper linkage to GoldSim.

Note: Arrays in C/C++ start at zero, rather than one (the default for FORTRAN). Array indices in this section use the C/C++ convention.

The arguments are :

```
int      XFmethod;  // Action that the external function
                    // must perform  (see table below)
int*     XFState;   // Returned value success 0 or fatal 1
double* inargs;     // Array of input arguments
double* outargs;    // This array returns different
                    // information when different XFmethod
                    // values are passed to the external
```

```
// function.
```

The values for `XFmethod are`:

0	XF_INITIALIZE	Initialize (called at the beginning of each realization). No arguments are passed on this call.
1	XF_CALCULATION	Normal calculation. Total number of output arguments are returned on this call. outargs[0] = 1st result from the function outargs[1] = 2nd result from the function etc.
2	XF_REP_VERSION	External functions report their versions. No input arguments are passed on this call. outargs[0] = external function version, e.g. 1.23
3	XF_REP_ARGUMENTS	External functions report the # of input and output arguments. No input arguments are passed on this call. outargs[0] = # of input arguments. outargs[1] = # of output arguments.
99	XF_CLEANUP	Close any open files, and optionally release dynamic memory at the end of a simulation. No arguments are passed on this call.

The returned values for XFState are:

0	OK, continue GoldSim
>0 and < 99	terminate GoldSim
99	OK, unload the DLL

The following two return codes may only be used for an XF_CALCULATION call:

-1	Fatal error, error message pointer returned
-2	More result memory required; the total amount (in doubles) required is returned in outargs[0]. This may be required of the external function is returning a table or time series definition. (Note that the additional memory is retained until the end of the simulation.)

The memory for the inargs and outargs arrays is allocated by GoldSim at the start of the simulation, based upon the inputs and outputs specified in the Interface tab of the External element properties dialog. The number of input and output arguments is verified during the XF_REP_ARGUMENTS request. GoldSim counts up the number of input and output arguments it expects, and compares it to the numbers for each reported by the DLL.

Warning: It is the responsibility of the external function to ensure that it ***does not*** write to any output arguments beyond the number reported for XF_REP_ARGUMENTS. Doing so may cause memory corruption in GoldSim.

Argument Checking

GoldSim calculates the total number of input and output arguments by summing over the the inputs and outputs specified on the Interface tab of the External element properties dialog. Note that each scalar input or output counts as one argument, while array inputs or outputs count as the size of the array (rows * columns). The calculated totals are then compared to the external function's reported number of input and output arguments. If the number of arguments defined by GoldSim does not agree with the number reported by the external function, GoldSim terminates with an error message.

However, note the following exceptions:

- If outargs[0] is returned as -1, the number of input arguments is not tested. This allows for functions where the DLL does not know in advance the number of input argument that it will receive.
- If the external function will be returning definitions for one or more Tables or Time Series (see below), GoldSim will not know in advance how long the Table definitions will be. In this case, the external function should specify a value for outargs[1] greater than or equal to the actual total number of arguments that may be returned. GoldSim will allocate this amount of memory for outargs. Note that this can be reset during the simulation by returning XFState=-2.

The Input and Output Argument Arrays

The content of input and output argument arrays is determined from the Interface tab of the External element properties dialog.

The following points should be noted:

- Input or outputs are processed in the order specified in the interface. All data from one input or output is contiguous, followed by the data from the next input or output.
- Scalar inputs and outputs are mapped to a single array argument.
- Vector input and output items are specified in order, one argument per item, according to the order specified in the array label.
- Matrix input and output items are specified item-by-item, with all items in one row together, followed by all items in the next row, and so on.
- Lookup Table definitions can be specified in the output interface, via a special format.
- Time Series definitions can be specified as inputs or outputs, also via a special format.

Lookup Table Definitions

External functions can also return Table definition elements to GoldSim. A table definition requires a specific sequence of values, depending whether it is a 1-D, 2-D, or 3-D table.

The sequence for a 1-D table is as follows:

- number of dimensions (in this case, 1)
- the number of rows

- row value1, row value 2, …, row value n
- dependent value 1, dependent value 2, …, dependent value n

The sequence for a 2-D table is as follows:

- number of dimensions (in this case, 2)
- the number of rows, the number of columns
- row value1, row value 2, …, row value n
- column value 1, column value 2, …, column value n
- dependent(row 1, column 1), …, dependent(row 1,column n)
- dependent(row 2, column 1), …, dependent(row 2,column n)
- …
- dependent(row n, column 1), …, dependent(row n, column n)

Warning: This is different than the sequence used to read in an ASCII text file for a 2-D table.

The sequence for a 3-D table is as follows:

- number of dimensions (must be 3)
- the number of rows, the number of columns, the number of layers
- row value1, row value 2, …, row value y
- column value 1, column value 2, …, column value x
- layer value1, layer value 2, …, layer value z
- dependent(row 1, column 1, layer 1), …, dependent(row 1,column x, layer 1)
- dependent(row 2, column 1, layer 1), …, dependent(row 2,column x, layer 1)
- …
- dependent(row y, column 1, layer 1), …, dependent(row y, column x, layer 1)
- .
  .
  .
- dependent(row 1, column 1, layer z), …, dependent(row 1,column x, layer z)
- dependent(row 2, column 1, layer z), …, dependent(row 2,column x, layer z)
- …
- dependent(row y, column 1, layer 1), …, dependent(row y, column x, layer z)

Warning: This is different than the sequence used to read in an ASCII text file for a 3-D table.

Time Series Definitions

External functions can also read and return Time Series Definition. A Time Series Definition consists of the following specific sequence of values.

1. The number 20 (this informs GoldSim that this is a Time Series)
2. The number -3 (this is a format number that infoms GoldSim what version of the time series format is expected)
3. Calendar-baed index: 0 if elapsed time; 1 if dates
4. An index (0,1,2,3) indicating what the data represents (0=instantaneous value, 1=constant value over the next time interval, 2=change over the next time interval, 3=discrete change)
5. The number of rows (0 for scalar time series)
6. The number of columns (0 for scalar and vector time series)
7. Number of series
8. For each series, the following is repeated:
   - The total number of time points in the series
   - Time point 1, Time point 2, …, Time point n

   The structure of the remainder of the file depends on whether the Time Series Definition represents a scalar, a vector, or a matrix.

   For a scalar, the next sequence of values is as follows:

   - Value 1[time point 1], Value 2[time point 2], …, Value[time point n]

   For a vector, the next sequence of values is as follows:

   - Value[row1, time point 1], Value[row1, time point 2], …, Value[row1, time point n]
   - Value[row2, time point 1], Value[row2, time point 2], …, Value[row2, time point n]
   - …
   - Value[row*r*, time point 1], Value[row*r*, time point 2], …, Value[row*r*, time point n]

   For a matrix, the next sequence of values is as follows:

   - Value[row1, column1, time point 1], Value[row1, column1, time point 2], …, Value[row1, column1, time point n]
   - Value[row1, column2, time point 1], Value[row1, column2, time point 2], …, Value[row1, column2, time point n]
   - …
   - Value[row1, column*c*, time point 1], Value[row1, column*c*, time point 2], …, Value[row1, column*c*, time point n]
   - .

- .
- .
- Value[row*r*, column1, time point 1], Value[row*r*, column1, time point 2], …, Value[row*r*, column1, time point n]
- Value[row*r*, column2, time point 1], Value[row*r*, column2, time point 2], …, Value[row*r*, column2, time point n]
- …
- Value[row*r*, column*c*, time point 1], Value[row*r*, column*c*, time point 2], …, Value[row*r*, column*c*, time point n]

External Function Examples

The following is a simple example implementation of DLL code that will work with an External element. This code takes two scalar elements as input, and returns the sum and product to GoldSim. For simplicity, this code is written in C, implemented with Visual C++. This external function can be used with the External.gsm example included with the GoldSim installation.

```
// Global enumerations, useful for C-style implementations
//
// XFMethodID identifies the method types, used to identify the phase of the
simulation
// that is currently in progress.
//
//    XF_INITIALIZE    - Called after DLL is loaded and before each
realization.
//    XF_CALCULATE     - Called during the simulation, each time the inputs
change.
//    XF_REP_VERSION   - Called after DLL load to report the external fcn
version number.
//    XF_REP_ARGUMENTS - Called after DLL load to report the number of input
and output //                         arguments.
//    XF_CLEANUP       - Called before the DLL is unloaded.
//
enum XFMethodID
{
    XF_INITIALIZE = 0, XF_CALCULATE = 1, XF_REP_VERSION = 2, XF_REP_ARGUMENTS =
3,
    XF_CLEANUP = 99
};

// XFStatusID identifies the return codes for external functions.
//
//     XF_SUCCESS          – Call completed successfully.
//     XF_CLEANUP_NOW      - Call was successful, but GoldSim should clean up
//                           and unload the DLL immediately.
//     XF_FAILURE          - Failure (no error information returned).
//     XF_FAILURE_WITH_MSG – Failure, with DLL-supplied error message
available.
//                           Address of error message is returned in the first
element
//                           of the output arguments array.
//     XF_INCREASE_MEMORY  - Failed because the memory allocated for output
arguments
//                           is too small.  GoldSim will increase the size of
the
//                           output argument array and try again.
//
```

```
enum XFStatusID
{
    XF_SUCCESS = 0, XF_FAILURE = 1, XF_CLEANUP_NOW = 99, XF_FAILURE_WITH_MSG =
-1,
    XF_INCREASE_MEMORY   = -2
};

////////////////////////////////////////////////////////////////////////////
// AddMultScalarsInC
//     Adds and multiplies two input scalar values (C Language implementation).
//--------------------------------------------------------------------------
extern "C" void __declspec(dllexport) AddMultScalarsInC(int     methodID,
                                                        int*    status,
                                                        double* inargs,
                                                        double* outargs)
{
    *status = XF_SUCCESS;
    switch ( methodID )
    {
    case  XF_INITIALIZE:
        break;                                  // nothing required

    case  XF_REP_VERSION:
        outargs[0] = 1.03;
        break;

    case  XF_REP_ARGUMENTS:
        outargs[0] = 2.0;                       // 2 scalar inputs expected
        outargs[1] = 2.0;                       // 2 scalar outputs returned
        break;

    case  XF_CALCULATE:
        outargs[0] = inargs[0] + inargs[1];   // return the sum
         outargs[1] = inargs[0]*inargs[1];     // return the product
        break;

    case  XF_CLEANUP:
        break;                                  // No clean-up required

    default:
        *status = XF_FAILURE;
        break;
    }
}
```

The following code is the same algorithm, implemented in Intel Visual Fortran.

```
! Utility module to specify the GoldSim parameter constants
module gs_parameters
    implicit none

    ! Parameters to identify the method types, which indicate the phase of the
    ! simulation that is currently in progress.
    !
    ! INITIALIZE       - Called after DLL is loaded and before each
realization.
    ! CALCULATE        - Called during the simulation, each time the inputs
change.
    ! REPORT_VERSION   - Called after DLL load to report the external fcn
```

```
version number.
    ! REPORT_ARGUMENTS - Called after DLL load to report the number of input
and output
    !                    arguments.
    ! CLEANUP          - Called before the DLL is unloaded.

    integer(4), parameter :: INITIALIZE       =  0
    integer(4), parameter :: CALCULATE        =  1
    integer(4), parameter :: REPORT_VERSION   =  2
    integer(4), parameter :: REPORT_ARGUMENTS =  3
    integer(4), parameter :: CLEANUP          = 99

    ! Parameters to identify the return codes for external functions.
    !
    ! SUCCESS          – Call completed successfully.
    ! CLEANUP_NOW      - Call was successful, but GoldSim should clean up
    !                    and unload the DLL immediately.
    ! FAILURE          - Failure (no error information returned).
    ! FAILURE_WITH_MSG – Failure, with DLL-supplied error message available.
    !                    Address of error message is returned in the first
element
    !                    of the output arguments array.
    ! INCREASE_MEMORY  - Failed because the memory allocated for output
arguments
    !                    is too small.  GoldSim will increase the size of the
                         output argument array and try again.

    integer(4), parameter :: SUCCESS          =  0
    integer(4), parameter :: FAILURE          =  1
    integer(4), parameter :: CLEANUP_NOW      = 99
    integer(4), parameter :: FAILURE_WITH_MSG = -1
    integer(4), parameter :: INCREASE_MEMORY  = -2

end module gs_parameters


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! add_mult_scalars
!     Adds and multiplies two input scalar values.
!---------------------------------------------------------------------
subroutine add_mult_scalars(method_id, status, inargs, outargs)
    !DEC$ ATTRIBUTES dllexport,c :: add_mult_scalars
    !DEC$ ATTRIBUTES value       :: method_id
    !DEC$ ATTRIBUTES reference   :: status
    !DEC$ ATTRIBUTES reference   :: inargs
    !DEC$ ATTRIBUTES reference   :: outargs

    use gs_parameters
    implicit none
    real(8),    parameter :: VERSION  = 1.03
    integer(4), parameter :: NINPUTS  = 2 ! Two scalar inputs expected
    integer(4), parameter :: NOUTPUTS = 2 ! Two scalar outputs returned
    integer(4) method_id, status
    real(8)    inargs(*), outargs(*)

    select case (method_id)
    case (INITIALIZE)
        status = SUCCESS
    case (REPORT_VERSION)
        outargs(1) = VERSION
        status = SUCCESS
    case (REPORT_ARGUMENTS)
        outargs(1) = NINPUTS
```

```
        outargs(2) = NOUTPUTS
        status = SUCCESS
    case (CALCULATE)
        outargs(1) = inargs(1) + inargs(2)      ! return the sum
        outargs(2) = inargs(1)*inargs(2)        ! return the product
        status = SUCCESS
    case (CLEANUP)
        status = SUCCESS
    case default
        status = FAILURE
    end select

end subroutine add_mult_scalars
```

Additional DLL code examples can be found (including examples for arrays, Lookup Tables, and Time Series Elements) can be found in the GoldSim install directory. In addition to the source files, example solution and project files for Microsoft Visual C++ and Intel Visual Fortran are also included (under General Examples/External).

External Function Calling Sequence

GoldSim makes calls to the DLL external function at different times during the course of a GoldSim simulation:

- Before the simulation starts (while checking model integrity).
- The first time that the External element values are calculated.
- Every time that the input values of the External element change.
- Before each (subsequent) realization.
- After each realization (Cleanup After Realization option only).
- After the simulation finishes.
- If any DLL external function call returns a value of 99.

GoldSim users can control when the DLL is unloaded via the Unload After Each Use and Cleanup After Realization options. These options are selected via checkboxes in the External element properties dialog ("Unload DLL after each use" and "Run Cleanup after each realization"). As the name implies, Unload After Each Use will clean up (XFMethod = XF_CLEANUP) and unload the DLL after each calculation call (XFMethod = XFCalculate). Similarly, Cleanup After Realization will clean up and unload the DLL at the end of each realization (if the DLL is loaded).

The DLL external function code can also control when the DLL is unloaded. If any call to a DLL external function returns a status of 99, GoldSim will treat the call as a success, but will clean up and unload the DLL immediately after processing the returned data.

Before the Simulation

GoldSim calls the DLL external function before the simulation starts, as part of the process to check the validity of the model. The main reason is to get the number of input and output arguments, to insure that they are consistent with what is specified in the Interface tab. The call sequence is as follows:

1. Load the DLL and check for the existence of the external function
2. Ask for the version number (XFMethod = XF_REP_VERSION).

3. Ask for the number of input and output arguments (XFMethod = XF_REP_ARGUMENTS), and compare to the values determined from the External element interface.
4. Clean up (XFMethod = XF_CLEANUP) and unload the DLL.

If any of these calls fail, or if the number of input or output arguments in step 3 are inconsistent, GoldSim will display an error message and return to the previous state (Edit or Ready mode).

During Each Realization

During each realization, GoldSim will call the DLL external function when the corresponding External element needs to be updated. This happens the first time that the Element is referenced, and every subsequent time-step or event update where an input to the Element changes. In this case, the following call sequence is used:

1. Check to see if the DLL is loaded. If so, skip to step 6.
2. Load the DLL and check for the existence of the external function
3. Ask for the version number (XFMethod = XF_REP_VERSION), and write it to the log file.
4. Ask for the number of input and output arguments (XFMethod = XF_REP_ARGUMENTS).
5. Initialize the DLL (XFMethod = XF_INITIALIZE).
6. Calculate (XFMethod = XF_CALCULATE)
7. If Unload After Each Use is specified, clean up (XFMethod = XF_CALCULATE) and unload the DLL.

Before Each Realization

Before each realization, GoldSim will check to see if the DLL is loaded for each External element. If the DLL is loaded, GoldSim will reinitialize the DLL (XFMethod = XF_INITIALIZE).

After Each Realization

If the Cleanup After Realization option is specified, and the DLL is loaded, GoldSim will clean up (XFMethod = XF_CLEANUP) and unload the DLL after each realization.

After the Simulation

After the simulation completes (either successfully or after a fatal error), GoldSim will clean up (XFMethod = XF_CLEANUP) and unload the DLL if it is still loaded.

DLL Calling Details

GoldSim can call DLL external functions by two different mechanisms, depending upon the Separate Process Space option. For each External element, the GoldSim user can select this option in the properties dialog, via the "Run in separate process space" checkbox. If this option is not selected, the DLL will be loaded with the Win32 LoadLibrary function. As a result, this DLL will share the same memory address space as the GoldSim process, and any memory used in the DLL will be charged to the GoldSim process. By default, External elements are created without the Separate Process Space option enabled.

If the Separate Process Space option is selected, GoldSim will call the DLL using a Component Object Model (COM) interface. The interface is implemented as a COM LocalServer executable, which is started when GoldSim first requests to load the DLL. Once the LocalServer is started, it loads the DLL into its own memory address space (separate from GoldSim), and acts a proxy between GoldSim and the DLL for all external function requests. After the DLL

is unloaded, GoldSim frees the COM interface and the LocalServer executable terminates. Because this option loads the DLL into a separate memory space, it may be a better option for DLLs with a large memory footprint.

64-Bit DLL Support

GoldSim also supports loading of external DLLs that are built as 64-bit libraries. The main advantage for a 64-bit DLL is a significant increase in the amount of virtual memory available (from 4GB to 8TB). Migrating DLL code from 32-bit to 64-bit requires minimal (if any) changes; just install the 64-bit compilers for Visual C++ or Visual Fortran and build with a 64-bit target. No change is GoldSim model configuration is required to use 64-bit DLLs, since GoldSim will automatically determine the type of DLL and call the appropriate interface. However, the following caveats do apply when using 64-bit DLLs in GoldSim:

- 64-bit DLLs can only be run on a computer with a 64-bit Windows operating system. If a DLL needs to run on both 32-bit and 64-bit Windows, it should be a 32-bit DLL (which will run on both 64-bit and 32-bit OS).
- 64-bit DLLs must run in a separate process space.
- For Distributed Processing runs, models that contain 64-bit DLLs can only be launched from a 64-bit Windows operating system. GoldSim will inspect the model to see if it contains a 64-bit DLL, and will disconnect all slaves that are running a 32-bit Windows OS.

Returning Error Messages from External Functions

To help GoldSim users debug problems with DLL external functions, GoldSim lets users send an error message from the DLL back to GoldSim through the External element interface when the call to an external function fails. The error message is then displayed to the user in a pop-up dialog.

The DLL external function signals the presence of an error message by returning a status value of -2. When GoldSim processes the results of the DLL external function call, it will interpret the first element of the output arguments arrray (outargs in our source-code example) as a **pointer** to a memory location where the error string can be found. The memory containing the string must have **static** scope, so that it will still be available when GoldSim retrieves the string. The string must also be NULL-terminated, even when returning from a FORTRAN DLL. If either of these recommendations are not followed, GoldSim will likely crash when it tries to display the error message.

The following code is an example of a C language function that will properly handle passing a message from a DLL external function to GoldSim. The ULONG_PTR is cast to different types on 64-bit (unsigned long) and 32-bit (unsigned __int64), so that it will work for building both 32-bit and 64-bit binaries.

```
// Utility method used to simplify sending an error message to GoldSim
void CopyMsgToOutputs(const char* sMsg, double* outargs)
{
    // Static character array used to hold the error message.
    // For the current incarnation of GoldSim, this is OK.
    // However, it will not be threadsafe if GoldSim simulations
    // become multithreaded.
    static char sBuffer[81];
    // Clear out any old data from the buffer
    memset(sBuffer, 0, sizeof(sBuffer));

    // Cast the first output array element as a pointer.
    // ULONG_PTR is used because it will work for both
      // 32-bit and 64-bit DLLs
```

```
    ULONG_PTR* pAddr = (ULONG_PTR*) outargs;

    // Copy the string data supplied into the static buffer.
    strncpy(sBuffer, sMsg, sizeof(sBuffer) - 1);

    // Copy the static buffer pointer to the first output array
    // element.
  *pAddr = (ULONG_PTR) sBuffer;
```

For FORTRAN DLLs, the following code performs the same function:

```
! Utility subroutine to simplify sending an error message to GoldSim
subroutine copy_msg_to_outputs(smsg, outargs)
    implicit none
    character(*) smsg
    real(8) outargs(*)
    ! "Static" character buffer, used to store the error message so
    ! that it can be returned to GoldSim.
    character(80), save :: sbuffer

    ! Create a shared memory variable that can be interpreted either as
    ! integer or real.
    integer(8) ioutput1
    real(8)    doutput1
    equivalence(ioutput1, doutput1)

    ! Copy the message into the buffer.  Truncate it if it is too long
    ! Make sure that it is null terminated!
    if (len(smsg) .lt. 80) then
        sbuffer = smsg // char(0)
    else
        sbuffer = smsg(1:79) // char(0)
    end if

    ! Since we are sending back to C++, we need an actual address.
    ! The "loc" function is not standard, but it is supported by all
    ! compilers that we checked.
    ioutput1 = loc(sbuffer)
    outargs(1) = doutput1
end subroutine copy_msg_to_outputs
```

Appendix D: GoldSim Units Database

364.4 Smoots plus 1 ear.

Official length of the Harvard Bridge

Appendix Overview

One of the more powerful features of GoldSim is that it is *dimensionally-aware*. You enable this capability by assigning dimensions to the outputs (and hence to the inputs) of the elements in your model. GoldSim has an extensive internal database of units and conversion factors. This appendix lists all of the units and conversion factors that are built into GoldSim.

Built-in Units and Conversion Factors

All units in GoldSim are defined relative to the following basic units:

- meter (m)
- kilogram (kg)
- second (s)
- Kelvin temperature (K)
- ampere (amp)
- radian (rad)
- Candela (cd)

The following table summarizes all of the built-in units and conversion factors within GoldSim, organized by category:

Unit	Abbreviation	Definition
Acceleration		
Mean Acceleration of earth's gravity	gee	9.80665 m/s2
Angle		
Degree of Arc	°	0.017453293 rad
One cycle/revolution	cycle	6.2831853 rad
Degree of arc	deg	0.017453293 rad
Minute of Arc	minarc	0.00029088821 rad
Radian	rad	1 rad
Revolution (cycle)	rev	6.2831853 rad
Second of Arc	secarc	4.8481368E-06 rad
Angular Frequency		
Revolutions per minute	pm	6°/s
Area		
Acre	ac	4046.856422 m2
Acre	acre	4046.856422 m2
Hectare	ha	10000 m2
Capacitance		
Farad	Fa	1 s4-amp2/kg-m2
Charge		
Coulomb of Charge	Co	1 s-amp
GigaCoulomb of Charge	GCo	1000000000 s-amp
KiloCoulomb of charge	kCo	1000 s-amp
MillaCoulomb of charge	mCo	0.001 s-amp
MegaCoulomb of charge	MCo	1000000 s-amp
NanoCoulomb of charge	nCo	1.00E-09 s-amp
PicoCoulomb of charge	pCo	1.00E-12 s-amp
TeraCoulomb of charge	TCo	1E+12 s-amp

Unit	Abbreviation	Definition
MicroCoulomb of charge	uCo	1.00E-06 s-amp
Currency		
US Dollar	$	(user defined)
Euro	EUR	(user defined)
British Pound	GBP	(user defined)
Japanese Yen	YEN	(user defined)
Australian Dollar	AUD	(user defined)
Brazilian Real	BRL	(user defined)
Canadian Dollar	CAD	(user defined)
Chinese Yuan	CNY	(user defined)
Czech Koruna	CZK	(user defined)
Danish Krona	DKK	(user defined)
Hong Kong Dollar	HKD	(user defined)
Hungarian Forint	HUF	(user defined)
Mexican Peso	MXN	(user defined)
New Zealand Dollar	NZD	(user defined)
Norwegian Krone	NOK	(user defined)
Russian Rouble	RUB	(user defined)
Singapore Dollar	SGD	(user defined)
Swedish Krona	SEK	(user defined)
Swiss Franc	CHF	(user defined)
South African Rand	ZAR	(user defined)
Thousand US Dollar	k$	1000 $
Thousand Euro	kEUR	1000 EUR
Thousand British Pound	kGBP	1000 GBP
Thousand Japanese Yen	kYEN	1000 YEN
Thousand Australian Dollar	kAUD	1000 AUD
Thousand Brazilian Real	kBRL	1000 BRL
Thousand Canadian Dollar	kCAD	1000 CAD
Thousand Chinese Yuan	kCNY	1000 CNY
Thousand Czech Koruna	kCZK	1000 CZK
Thousand Danish Krone	kDKK	1000 DKK
Thousand Hong Kong Dollar	kHKD	1000 HKD
Thousand Hungarian Forint	kHUF	1000 HUF
Thousand Mexican Peso	kMXN	1000 MZN
Thousand New Zealand Dollar	kNZD	1000 NZD
Thousand Norwegian Krone	kNOK	1000 NOK
Thousand Russian Rouble	kRUB	1000 RUB
Thousand Singapore Dollar	kSGD	1000 SGD
Thousand Swedish Krona	kSEK	1000 SEK
Thousand Swiss Franc	kCHF	1000 CHF

Unit	Abbreviation	Definition
Thousand South African Rand	kZAR	1000 ZAR
Million US Dollar	M$	1000000 $
Million Euro	MEUR	1000000 EUR
Million British Pound	MGBP	1000000 GBP
Million Japanese Yen	MYEN	1000000 YEN
Million Australian Dollar	MAUD	1000000 AUD
Million Brazilian Real	MBRL	1000000 BRL
Million Canadian Dollar	MCAD	1000000 CAD
Million Chinese Yuan	MCNY	1000000 CNY
Million Czech Koruna	MCZK	1000000 CZK
Million Danish Krone	MDKK	1000000 DKK
Million Hong Kong Dollar	MHKD	1000000 HKD
Million Hungarian Forint	MHUF	1000000 HUF
Million Mexican Peso	MMXN	1000000 MZN
Million New Zealand Dollar	MNZD	1000000 NZD
Million Norwegian Krone	MNOK	1000000 NOK
Million Russian Rouble	MRUB	1000000 RUB
Million Singapore Dollar	MSGD	1000000 SGD
Million Swedish Krona	MSEK	1000000 SEK
Million Swiss Franc	MCHF	1000000 CHF
Million South African Rand	MZAR	1000000 ZAR
Current		
Ampere	amp	1 amp
GigaAmpere	Gamp	1000000000 amp
KiloAmpere	kamp	1000 amp
MilliAmpere	mamp	0.001 amp
MegaAmpere	Mamp	1000000 amp
NanoAmpere	namp	1.00E-09 amp
PicoAmpere	pamp	1.00E-12 amp
TeraAmpere	Tamp	1E+12 amp
MicroAmpere	uamp	1.00E-06 amp
Dose		
GigaGray (dose absorbed)	GGy	1000000000 m2/s2
GigaSievert (dose equivalent)	GSv	1000000000 m2/s2
Gray (dose absorbed)	Gy	1 m2/s2
KiloGray (dose absorbed)	kGy	1000 m2/s2
KiloSievert (dose equivalent)	kSv	1000 m2/s2
MilliGray (dose absorbed)	mGy	0.001 m2/s2
MegaGray (dose absorbed)	MGy	1000000 m2/s2
MilliSievert (dose equivalent)	mSv	0.001 m2/s2
MegaSievert (dose equivalent)	MSv	1000000 m2/s2

Unit	Abbreviation	Definition
NanoGray (dose absorbed)	nGy	1.00E-09 m2/s2
NanoSievert (dose equivalent)	nSv	1.00E-09 m2/s2
PicoGray (dose absorbed)	pGy	1.00E-12 m2/s2
PicoSievert (dose equivalent)	pSv	1.00E-12 m2/s2
RAD dose	RADD	0.01 m2/s2
REM dose	REM	0.01 m2/s2
MilliREM dose	mREM	1.00E-05 m2/s2
Sievert (dose equivalent)	Sv	1 m2/s2
TeraGray (dose absorbed)	TGy	1E+12 m2/s2
TeraSievert (dose equivalent)	TSv	1E+12 m2/s2
MicroGray (dose absorbed)	uGy	1.00E-06 m2/s2
MicroSievert (dose equivalent)	uSv	1.00E-06 m2/s2
Electrical Resistance		
GigaOhm	Gohm	1000000000 kg-m2/s3-amp2
KiloOhm	kohm	1000 kg-m2/s3-amp2
MilliOhm	mohm	0.001 kg-m2/s3-amp2
MegaOhm	Mohm	1000000 kg-m2/s3-amp2
PicoOhm	pohm	1.00E-12 kg-m2/s3-amp2
NanoOhm	nohm	1.00E-09 kg-m2/s3-amp2
Ohm	ohm	1 kg-m2/s3-amp2
TeraOhm	Tohm	1E+12 kg-m2/s3-amp2
MicroOhm	uohm	1.00E-06 kg-m2/s3-amp2
Energy		
British Thermal Unit (Int'l)	BTU	1055.056 kg-m2/s2
Calorie	cal	4.1868 kg-m2/s2
Electron Volt	eV	1.60E-19 kg-m2/s2
GigaCalorie	Gcal	4186800000 kg-m2/s2
GigaElectron volt	GeV	1.60E-10 kg-m2/s2
GigaJoule	GJ	1000000000 kg-m2/s2
Joule	J	1 kg-m2/s2
KiloCalorie	kcal	4186.8 kg-m2/s2
KiloElectron volt	keV	1.60E-16 kg-m2/s2
KiloJoule	kJ	1000 kg-m2/s2
Kilowatt-hour	kwh	3600000 kg-m2/s2
MilliCalorie	mcal	0.0041868 kg-m2/s2
MegaCalorie	Mcal	4186800 kg-m2/s2
MilliElectron volt	meV	1.60E-22 kg-m2/s2
MegaElectron volt	MeV	1.60E-13 kg-m2/s2
MilliJoule	mJ	0.001 kg-m2/s2
MegaJoule	MJ	1000000 kg-m2/s2

Unit	Abbreviation	Definition
NanoCalorie	ncal	4.19E-09 kg-m2/s2
NanoElectron volt	neV	1.60E-28 kg-m2/s2
NanoJoule	nJ	1.00E-09 kg-m2/s2
PicoCalorie	pcal	4.19E-12 kg-m2/s2
PicoElectron volt	peV	1.60E-31 kg-m2/s2
PicoJoule	pJ	1.00E-12 kg-m2/s2
TeraCalorie	Tcal	4.1868E+12 kg-m2/s2
TeraElectron volt	TeV	1.60E-07 kg-m2/s2
TeraJoule	TJ	1E+12 kg-m2/s2
MicroCalorie	ucal	4.19E-06 kg-m2/s2
MicroElectron volt	ueV	1.60E-25 kg-m2/s2
MicroJoule	uJ	1.00E-06 kg-m2/s2
Flux (Volume)		
Acre-feet/day	afd	0.01427641 m3/s
US Barrels/day	bpd	1.84E-06 m3/s
Cubic feet per second	cfs	0.028316847 m3/s
US Gallons per minute	gpm	6.31E-05 m3/s
Million gallons per day	MGD	0.043812639 m3/s
Force		
Dyne	dyne	1.00E-05 kg-m/s2
GramForce	gf	0.00980665 kg-m/s2
GigaNewton	GN	1000000000 kg-m/s2
Kilogram Force	kgf	9.80665 kg-m/s2
1,000 Pound force	kip	4448.221909 kg-m/s2
KiloNewton	kN	1000 kg-m/s2
Pound force	lbf	4.448221909 kg-m/s2
Milligram force	mgf	9.81E-06 kg-m/s2
MilliNewton	mN	0.001 kg-m/s2
MegaNewton	MN	1000000 kg-m/s2
Newton	N	1 kg-m/s2
NanoNewton	nN	1.00E-09 kg-m/s2
Ounce force	ozf	0.278013869 kg-m/s2
PicoNewton	pN	1.00E-12 kg-m/s2
TeraNewton	TN	1E+12 kg-m/s2
Ton force	tonf	8896.443819 kg-m/s2
MicroNewton	uN	1.00E-06 kg-m/s2
Frequency (non-angular, Rate)		
PicoHertz (frequency)	pHz	1.00E-12 1/s
Becquerel	Bq	1 1/s
Curie	Ci	37000000000 1/s
GigaBecquerel	GBq	1000000000 1/s

Unit	Abbreviation	Definition
GigaHertz (frequency)	GHz	1000000000 1/s
Hertz (frequency)	Hz	1 1/s
KiloBecquerel	kBq	1000 1/s
KiloHertz (frequency)	kHz	1000 1/s
MilliBecquerel	mBq	0.0011/s
MegaBecquerel	MBq	1000000 1/s
MilliHertz (frequency)	mHz	0.001 1/s
MegaHertz (frequency)	MHz	1000000 1/s
NanoBacquerel	nBq	1.00E-09 1/s
NanoHertz (frequency)	nHz	1.00E-09 1/s
PicoBecquerel	pBq	1.00E-12 1/s
Picocurie	pCi	0.037 1/s
TeraBacquerel	TBq	1E+12 1/s
TeraHertz (frequency)	THz	1E+12 1/s
MicroBecquerel	uBq	1.00E-06 1/s
MicroCurie	uCi	37000 1/s
MicroHertz (frequency)	uHz	1.00E-06 1/s
Illuminance		
Lambert	lamb	10000 cd/m2
Lux (1 lm/m^2)	lx	1 cd/m2
Inverse Area		
Miles per gallon	mpg	425143.68321/m2
Items		
Item	item	(dimensionless)
Persons	pers	1 item
Thousand Persons	kpers	1000 item
Million Persons	Mpers	1000000 item
Length		
Angstrom	Ang	1.00E-10 m
Centimeter	cm	0.01 m
Fathom	fath	1.8288 m
Furlong	flng	201.168 m
Foot (US)	ft, '	0.3048 m
GigaMeter	Gm	1000000000 m
Inch	in, "	0.0254 m
KiloMeter	km	1000 m
Light Year	ly	9.46E+15 m
Meter	m	1 m
Mile	mi	1609.344 m
Mil=0.001 inch	mil	2.54E-05 m
MilliMeter	mm	0.001 m

Unit	Abbreviation	Definition
MegaMeter	Mm	1000000 m
Nautical Mile	naut	1852 m
NanoMeter	nm	1.00E-09 m
PicoMeter	pm	1.00E-12 m
Rod	rd	5.0292 m
TeraMeter	Tm	1E+12 m
MicroMeter	um	1.00E-06 m
Yard	yard	0.9144 m
Luminous Intensity		
Candela	cd	1 cd
GigaCandela	Gcd	1000000000 cd
KiloCandela	kcd	1000 cd
Lumen (cd/Sterradian)	lm	0.079577472 cd
MilliCandela	mcd	0.001 cd
NanoCandela	ncd	1.00E-09 cd
PicoCandela	pcd	1.00E-12 cd
TeraCandela	Tcd	1E+12 cd
MicroCandela	ucd	1.00E-06 cd
MegaCandela	Mcd	1000000 cd
Mass		
Gram	g	0.001 kg
GigaGram	Gg	1000000 kg
KiloGram	kg	1 kg
Pound (mass)	lbm	0.4535924 kg
MilliGram	mg	1.00E-06 kg
MegaGram	Mg	1000 kg
NanoGram	ng	1.00E-12 kg
Ounce (mass)	ozm	0.028349525 kg
PicoGram	pg	1.00E-15 kg
Slug Mass	slug	14.5939039 kg
TeraGram	Tg	1000000000 kg
Ton (mass)	tonm	907.1848 kg
Tonne	tonne	1000 kg
MicroGram	ug	1.00E-09 kg
Math Constants		
Parts per billion	ppb	1.00E-09
Parts per million	ppm	1.00E-06
Percentage	%	0.01
Permeability (seepage)		
Darcy	Darcy	9.87E-13 m2
Millidarcy	md	9.87E-16 m2

Unit	Abbreviation	Definition
Power		
GigaWatt	GW	1000000000 kg-m2/s3
Horsepower (550 ft-lb/sec)	hp	745.6999209 kg-m2/s3
KiloWatt	kW	1000 kg-m2/s3
MilliWatt	mW	0.001 kg-m2/s3
MegaWatt	MW	1000000 kg-m2/s3
NanoWatt	nW	1.00E-09 kg-m2/s3
PicoWatt	pW	1.00E-12 kg-m2/s3
TeraWatt	TW	1E+12 kg-m2/s3
MicroWatt	uW	1.00E-06 kg-m2/s3
Watt	W	1 kg-m2/s3
Pressure, Stress		
Atmosphere	atm	101325 kg/m-s2
Bar	bar	100000 kg/m-s2
GigaBar	Gbar	1E+14 kg/m-s2
GigaPascal	GPa	1000000000 kg/m-s2
KiloBar	kbar	100000000 kg/m-s2
KiloPond	kp	98066.5 kg-m/s2
KiloPascal	kPa	1000 kg/m-s2
MilliBar	mbar	100 kg/m-s2
MegaBar	Mbar	1E+11 kg/m-s2
MilliPascal	mPa	0.001 kg/m-s2
MegaPascal	MPa	1000000 kg/m-s2
NanoBar	nbar	0.0001 kg/m-s2
NanoPascal	nPa	1.00E-09 kg/m-s2
Pascal	Pa	1 kg/m-s2
PicoBar	pbar	1.00E-07 kg/m-s2
PicoPascal	pPa	1.00E-12 kg/m-s2
Pound per square foot	psf	47.88026215 kg/m-s2
Pound per square inch	psi	6894.757749 kg/m-s2
TeraBar	Tbar	1.00E+17 kg/m-s2
Torr (mm Hg)	torr	133.3221913 kg/m-s2
TeraPascal	TPa	1E+12 kg/m-s2
MicroBar	ubar	0.1 kg/m-s2
MicroPascal	uPa	1.00E-06 kg/m-s2
Quantity of Matter		
GigaMol	Gmol	1000000000 mol
KiloMole	kmol	1000 mol
MilliMole	mmol	0.001 mol
MegaMole	Mmol	1000000 mol
Mole	mol	1 mol

Unit	Abbreviation	Definition
NanoMole	nmol	1.00E-09 mol
PicoMole	pmol	1.00E-12 mol
TeraMole	Tmol	1E+12 mol
MicroMole	umol	1.00E-06 mol
Temperature		
Celsius temperature	C	1 K
Celsius degree	Cdeg	1 K
Fahrenheit temperature	F	0.555555556 K
Fahrenheit Degree	Fdeg	0.555555556 K
GigaKelvin temperature	GK	1000000000 K
Kelvin temperature	K	1 K
KiloKelvin temperature	kK	1000 K
MilliKelvin temperature	mK	0.001 K
MegaKelvin temperature	MK	1000000 K
NanoKalvin temperature	nK	1.00E-09 K
PicoKelvin temperature	pK	1.00E-12 K
Rankine temperature	R	0.555555556 K
TeraKelvin temperature	TK	1E+12 K
MicroKelvin temperature	uK	1.00E-06 K
Time		
Year	a, yr	31557600 s
Day	d, day	86400 s
GigaSeconds of time	Gs	1000000000 s
Hour	hr	3600 s
Kilosecond of time	ks	1000 s
Minute of time	min	60 s
Month	mon	2629800 s
MilliSecond of time	ms	0.001 s
MegaSecond of time	Ms	1000000 s
NanoSecond of time	ns	1.00E-09 s
PicoSecond of time	ps	1.00E-12 s
Second of time	s, sec	1 s
TerraSecond of time	Ts	1E+12 s
MicroSecond of time	us	1.00E-06 s
Week	week	604800 s
Date	date	86400 s
Date and time	datetime	86400 s
Velocity		
Feet per minute	fpm	0.00508 m/s
Feet per second	fps	0.3048 m/s
Kilometer per hour	kph	0.277777778 m/s

Unit	Abbreviation	Definition
Knots	kt	0.514444444 m/s
Miles per hour	mph	0.44704 m/s
Viscosity (Absolute)		
Centipose	cp	0.001 kg/m/s
Poise	poise	0.1 kg/m/s
Viscosity (Kinematic)		
Stoke	stoke	0.0001 m2/s
Voltage		
GigaVolt	GV	1000000000 kg-m2/s3-amp
KiloVolt	kV	1000 kg-m2/s3-amp
MilliVolt	mV	0.001 kg-m2/s3-amp
MegaVolt	MV	1000000 kg-m2/s3-amp
NanoVolt	nV	1.00E-09 kg-m2/s3-amp
PicoVolt	pV	1.00E-12 kg-m2/s3-amp
TeraVolt	TV	1E+12 kg-m2/s3-amp
MicroVolt	uV	1.00E-06 kg-m2/s3-amp
Volt	V	1 kg-m2/s3-amp
Volume		
Acre-feet	af	1233.48183754752 m3
Thousand Acre-feet	kaf	1233481.83754752 m3
Million Acre-feet	Maf	1233481837.54752 m3
Barrel (US, oil)	bbl	0.1589873 m3
Barrel (US, dry)	bbldry	0.11563 m3
Barrel (US, liquid)	bblliq	0.11924 m3
Bushel	bushel	0.03523907 m3
Cubic Centimeter	cc	1.00E-06 m3
Cup, US	cup	0.000236588 m3
Fluid ounce	floz	2.96E-05 m3
Gallon (US)	gal	0.003785412 m3
Gallon (Imperial)	gali	0.00454609 m3
Gallon (US)	galus	0.003785412 m3
GigaLitre	Gl, GL	1000000 m3
KiloLitre	kl, kL	1 m3
Litre	l, L	0.001 m3
MilliLitre	ml, mL	1.00E-06 m3
MegaLitre	Ml , ML	1000 m3
NanoLitre	nl	1.00E-12 m3
Pint, US liquid	pint	0.000473177 m3
PicoLitre	pl, pL	1.00E-15 m3
Quart (US)	qt	0.000946353 m3

Unit	Abbreviation	Definition
Standard cubic foot	stcf	0.028316847 m3
Tablespoon	tbsp	1.48E-05 m3
TeraLitre	Tl, TL	1000000000 m3
Teaspoon	tsp	4.93E-06 m3
MicroLitre	ul, uL	1.00E-09 m3

Appendix E: Database Input File Formats

Art and science cannot exist but in minutely organized particulars.

William Blake, To the Public

Appendix Overview

In simulations which require a great deal of input, it may be desirable that the simulation model can access the various data sources directly to ensure the quality of the data transfer.

To facilitate this, GoldSim data entry elements can be linked directly to an ODBC-compliant database.

Read more: Linking Elements to a Database (page 972).

After defining the linkage, you can then instruct GoldSim to download the data at any time. When it does this, *GoldSim internally records the time and date at which the download occurred*, along with other reference information retrieved from the database (e.g., document references), and this is stored with the model in the Run Log. This allows you to confirm that the correct data were loaded into your model, and provides very strong and defensible quality control over your model input data.

GoldSim can import from three different types of databases: a Generic Database, a Simple GoldSim Database, and a Yucca Mountain Database. This appendix describes the details of the structure and format for each of these three database types.

In this Appendix

This appendix discusses the following:

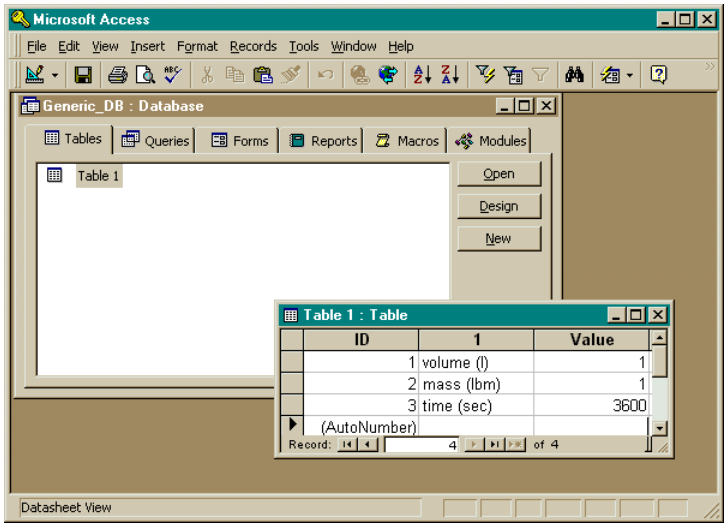
- Creating a Generic Database
- Creating a Simple GoldSim Database
- Creating a Yucca Mountain Database

Creating a Generic Database

The generic database format requirements are very general:

- The database must be ODBC compliant;
- The selected table must have a field (column) which contains unique IDs which will be used to map data to specific GoldSim elements. These IDs would typically be the GoldSim element names (but they do not have to be).
- The table must have one or more columns containing the data items to be downloaded.

The following figure shows a Microsoft Access screen-display when editing a simple generic database.



Note that by assigning multiple records and using multiple fields for each ID, you can download vector and matrix data from the generic database.

Read more: Downloading from a Generic Database (page 975).

The Database.gsm file in the General Examples folder in your GoldSim directory provides an illustration of how this can be done. This folder also includes the sample database referenced by this file (Generic_DB.accdb), created using Microsoft Access 2003. In order to use the GoldSim file, you will need to add the database as data sources to your computer.

Read more: Adding Data Sources to Your Computer (page 973).

Creating a Simple GoldSim Database

A Simple GoldSim database contains the following three tables:

- tbl_Parameter
- tbl_Parameter_Reference
- tbl_Probability_Value_Pairs

Each of these tables is discussed in detail below.

Parameter Table

The Parameter Table (tbl_Parameter) must contain the following fields (which must be named as shown below):

Field Name	Type	Description
UID	AutoNumber	Unique integer number assigned to each element.
Parameter_Name	Text	The ID of the GoldSim element. Note that the length of an element ID in GoldSim is limited to 30 characters.
Parameter_Path	Memo	The full path to the element in GoldSim. This attribute is optional. GoldSim tries to find a record set in the database using the name and path information. If this query is not successful it will try again just querying the name. If a path is specified it must end with '\'.
Type_Code	Text	Code to describe the parameter type (see below).
ModDate	Date/Time	Last Date that the parameter properties were changed. Note that this field is for information purposes only and is not used by GoldSim.
Current	Yes/No	This version of the parameter (this record) is considered active in GoldSim if this flag is set to Yes. Note that only one parameter with the same name (and path) should have the current flag set to true.
Description	Text	Description of the parameter, inserted into the Description field for the element.
Unit	Text	Unit abbreviation for the parameter. See Appendix D for unit abbreviations. The unit should be specified without parentheses or brackets.
Arg_1	Number (double)	Value for the first argument for the parameter. All parameters have at least one argument.
Arg_2	Number (double)	Value for the second argument for the parameter
Arg_3	Number (double)	Value for the third argument for the parameter
Arg_4	Number (double)	Value for the fourth argument for the parameter

Note: Particular care must be taken when importing Stochastics from a database, as different Stochastics could have different dimensions (e.g., a Binomial is always dimensionless; a Boolean is always a condition). If a Stochastic is defined in your model as a paticular type of distribution whose output is inconsistent with that of the distribution that you are trying to import from the database, GoldSim will display an error.

Note: Care must also be taken when importing Stochastics that represent temperatures. This is because temperatures have an absolute reference (i.e., absolute zero). *Differences* between temperatures are expressed in 'deg' units (Cdeg, Fdeg). This can lead to errors when importing Stochastics, since GoldSim assumes a single unit for the distribution parameters, while some types of distributions would require two different types of units. For example, a Normal distribution representing a temperature would require the Mean to be specifed in absolute units (e.g., C) and the Standard Deviation to be specified in difference units (e.g., Cdeg). If faced with this problem, there are several approaches for addressing this: 1) Use a distribution type where all parameters have the same units (e.g., triangular, uniform, cumulative); or 2) Specify and import the distributions as dimensionless and then apply a unit to the sampled value using an Expression element.

Read more: Dealing with Temperature Units (page 97).

Warning: The special GoldSim units "date" and "datetime" cannot be used when importing from a database.

Note that the Parameter_Path does not need to be defined (it can be blank). In this case, it does not matter where the element exists in the model. If you specify a path, it must start and end with a backslash (e.g., \container1\container2\ ). If you want to specify the top-level Container (the model root), you should use a single backslash.

If you specify a path, GoldSim first tries to find a record with a the specified element name and path. If it does not find it, it tries again, ignoring the path.

If GoldSim finds multiple records with the same name and path, and both have the Current field marked "Yes", it will issue an error message (i.e., if multiple records in the database have the same name and path, only one record can have the Current flag set to Yes).

Parameter Type Codes and Arguments

The codes for the various parameter types, and their required arguments are shown below:

Distribution	Type Code	Arg_1	Arg_2	Arg_3	Arg_4
constant (Data element)	100	Value			
constant (vector Data element)	101	Number of rows	1 (Number of columns)		
constant (matrix Data element)	102	Number of rows	Number of columns		
uniform	2100	Minimum	Maximum		
log-uniform	2101	Minimum	Maximum		
normal	2200	Mean	Std. Deviation		
truncated normal	2202	Mean	Std. Deviation	Minimum	Maximum

Distribution	Type Code	Arg_1	Arg_2	Arg_3	Arg_4
log-normal (geometic input)	2300	Geometric Mean	Geometric Std. Deviation		
truncated log-normal (geometric input)	2302	Geometric Mean	Geometric Std. Deviation	Minimum	Maximum
log-normal (true mean input)	2330	True Mean	True Std. Deviation		
truncated log-normal (true mean input)	2332	True Mean	True Std. Deviation	Minimum	Maximum
triangular	2400	Minimum	Most Likely	Maximum	
log triangular	2401	Minimum	Most Likely	Maximum	
triangular (10/90)	2402	10^{th} percentile	Most Likely	90^{th} percentile	
log triangular (10/90)	2403	10^{th} percentile	Most Likely	90^{th} percentile	
cumulative	2500	references Probability_Value_Pairs table (see below)			
Log-cumulative	2501	references Probability_Value_Pairs table (see below)			
discrete	2600	references Probability_Value_Pairs table (see below)			
poisson	2700	Expected Value			
beta (generalized)	2800	Mean	Std. Deviation	Minimum	Maximum
beta (success, failure)	2804	Sucesses	Failures		
BetaPERT	4200	Minimum	Most Likely	Maximum	
BetarPERT 10/90	4201	10^{th} percentile	Most Likely	90^{th} percentile	
gamma	2900	Mean	Std. Deviation		
truncated gamma	2902	Mean	Std. Deviation	Minimum	Maximum
weibull	3000	Minimum	Weibull Slope	Mean-Minimum	
truncated weibull	3002	Minimum	Weibull Slope	Mean-Minimum	Maximum

Distribution	Type Code	Arg_1	Arg_2	Arg_3	Arg_4
binomial	3100	# of Picks (Batch size)	Probability of Success		
boolean	3200	Probability of True			
Student's t	3300	Degrees of freedom			
exponential	3400	Mean			
pareto	3500	a	b		
truncated Pareto	3502	a	b	Maximum	
negative binomial	3600	Successes	Probability of Success		
extreme value (minimum)	3800	Location	Scale		
extreme value (maximum)	3803	Location	Scale		
extreme probability (minimum)	3900	Number of Samples			
extreme probability (maximum)	3903	Number of Samples			
Pearson type III	4000	Location	Scale	Shape	
sampled results (no extrapolation)	4100	references Probability_Value_Pairs table (see below)			
sampled results (extrapolation)	4103	references Probability_Value_Pairs table (see below)			

Note that for the Discrete, Cumulative and Sampled Results distributions, the first argument references the *val_pair_UID* field in the Probability Value Pairs table (described below).

Parameter Reference Table

The Parameter Reference Table (tbl_Parameter_Reference) allows you to specify reference information for the element.

The table has the following fields:

Field Name	Type	Description
Parameter_UID	Text	Primary Key – link from UID in Parameter Table
GS_Parameter_Note	Text	The text in this field overwrites the Note associated with the element in GoldSim. If left blank here, the Note in GoldSim is not overwritten.

When GoldSim imports text from a database into a GoldSim Note, it automatically converts text to hyperlinks in the Note under the following circumstances:

- Any text beginning the prefixes listed below is converted to a hyperlink. The hyperlink terminates when a space in encountered. As a result, hyperlinks with spaces will not be recognized by GoldSim.
  - http://
  - www.
  - ftp://
  - ftp.
  - file://
- If the character @ is encountered, and text before and after the @ not delimited by a space or a line break will be considered part of the hyperlink.

Array Values Table

The Array Values Table (tbl_Array_Values) is used to store an arbitrary number of array values for a vector or matrix Data element.

The table has the following fields:

Field Name	Type	Description
ID	Number	Unique integer number assigned to each row in the table.
Array_UID	Number	The parameter ID from the Parameter Table (tbl_Parameter) identifying the Data element.
Value	Number	The value for the specified row and column of the array.
Row	Number	The row of the array. This index is 1-based.
Column	Number	The column of the array. This index is 1-based. It must be set to 1 for vectors.

Probability Value Pairs Table

The Probability Value Pair Table (tbl_Probability_Value_Pairs) is used to store an arbitrary number of value pairs used in defining discrete, cumulative and sampled results distributions. No other type of element uses this table.

The table has the following fields:

Field Name	Type	Description
Val_pair_UID	Number	Identifies a set of value-probability-pairs that are used by discrete, cumulative and sampled results distributions. This distribution must specify this ID in arg_1 in tbl_Parameter.
Probability	Number	The probability (for discrete distributions) or cumulative probability (for Cumulative distributions) of the data pair (dimensionless). Ignored for sampled results distributions.

Field Name	Type	Description
Value	Number	The value corresponding with the defined probability level for Discrete, Cumulative and Sampled Results distributions. Uses the unit defined in the parameter table.

Example File and Database Template

The Database.gsm file in the General Examples folder in your GoldSim directory provides an example of how elements can be linked to a Simple GoldSim database. This folder also includes the sample database referenced by this file (Simple_DB.accdb), created using Microsoft Access 2003. In order to use the GoldSim file, you will need to add the database as data sources to your computer.

Read more: Adding Data Sources to Your Computer (page 973).

The General Examples folder also includes a template for creating Simple GoldSim databases (simple_db_template.accdb). This template file includes two additional tables (providing parameter type codes and unit abbreviations) which can be used to support the implementation of custom forms which allow the user to pick a parameter type code and unit from a list.

Creating a Yucca Mountain Database

A Yucca Mountain database must contain the following three tables:

- GS_Parameter
- GS_Parameter_Value
- GS_Value_Component

Each table is described in detail below.

Note: The tables described below can contain additional fields not used by GoldSim. When importing information, GoldSim will ignore any extra fields.

Parameter Table

The Parameter Table (GS_Parameter) has one record for each linked Element. It contains basic descriptive information about the Element, and an index (Parameter_ID) which links it into the GS_Parameter_Value table. It must contain the following fields:

No.	Field	Notes
1	Parameter_ID	Unique integer number assigned to each element
2	Parameter_Name	Key field which must match the GoldSim element ID
3	Description	Text description of the element
4	Units	String with GoldSim abbreviations for units of the data (see Appendix D for correct unit abbreviations)
5	Parameter_Code	100 Data element 1nn Array Data element, where nn is the # of columns (01 for vectors) 2100 Stochastic: uniform

No.	Field	Notes
		2101 Stochastic: log-uniform 2200 Stochastic: normal 2202 Stochastic: truncated normal 2300 Stochastic: lognormal (geometric mean) 2302 Stochastic: truncated lognormal (geometric mean) 2330 Stochastic: lognormal (true mean) 2332 Stochastic: truncated lognormal (true mean) 2400 Stochastic: triangular 2401 Stochastic: log-triangular 2402 Stochastic: triangular (10/90) 2403 Stochastic: log-triangular (10/90) 2500 Stochastic: cumulative 2501 Stochastic: Log-cumulative 2600 Stochastic: discrete: 2700 Stochastic: Poisson 2800 Stochastic: Generalized Beta 2804 Stochastic: Beta (Success, Failure) 2900 Stochastic: Gamma 2902 Stochastic: Truncated Gamma 3000 Stochastic: Weibull 3002 Stochastic: Truncated Weibull 3100 Stochastic: Binomial 3200 Stochastic: Boolean 3300 Stochastic: Student's t 3400 Stochastic: Exponential 3500 Stochastic: Pareto 3502 Stochastic: Truncated Pareto 3600 Stochastic: Negative Binomial 3800 Stochastic: extreme value (minimum) 3803 Stochastic: extreme value (maximum) 3900 Stochastic: extreme probability (minimum) 3903 Stochastic: extreme probability (maximum) 4000 Stochastic: Pearson type III 4100 Stochastic: sampled results 4103 Stochastic: sampled results (extrapolation) 4200 Stochastic: BetaPERT 4201 Stochastic: BetaPERT (10/90) 5100 1-D Table 52nn 2-D Table, where nn is the no. of columns

No.	Field	Notes
		File element (no code required)
6	indep_row_units	String with GoldSim abbreviations for units of a table element's Row independent variable (only required for tables)
7	indep_col_units	String with GoldSim abbreviations for units of the 2-D table element's Column independent variable (only required for 2-D tables)
8	bc_date	The default effective date for this item, in format YYYY-MM-DD HH:MM:SS. If no effective date is specified when downloading data, this date is used for this particular element.

Note: Particular care must be taken when importing Stochastics from a database, as different Stochastics could have different dimensions (e.g., a Binomial is always dimensionless; a Boolean is always a condition). If a Stochastic is defined in your model as a paticular type of distribution whose output is inconsistent with that of the distribution that you are trying to import from the database, GoldSim will display an error.

Note: Care must also be taken when importing Stochastics that represent temperatures. This is because temperatures have an absolute reference (i.e., absolute zero). *Differences* between temperatures are expressed in 'deg' units (Cdeg, Fdeg). This can lead to errors when importing Stochastics, since GoldSim assumes a single unit for the distribution parameters, while some types of distributions would require two different types of units. For example, a Normal distribution representing a temperature would require the Mean to be specifed in absolute units (e.g., C) and the Standard Deviation to be specified in difference units (e.g., Cdeg). If faced with this problem, there are several approaches for addressing this: 1) Use a distribution type where all parameters have the same units (e.g., triangular, uniform, cumulative); or 2) Specify and import the distributions as dimensionless and then apply a unit to the sampled value using an Expression element.

Read more: Dealing with Temperature Units (page 97).

Warning: The special GoldSim units "date" and "datetime" cannot be used when importing from a database.

Parameter Value Table

The Parameter Value Table (GS_Parameter_Value) has one record for each Element and each effective date. Each record must have a unique Effective_Date and Value_ID. The Value_ID is an index which is used to link into the actual data values, which are stored in table GS_Value_Component. The GS_Parameter_Value table must contain the following fields:

No.	Field	Notes
1	Parameter_ID	Key to link from items in table GS_Parameter
2	Value_ID	Unique integer key for value record(s) in table GS_Value_Component
3	Effective_Date	The effective date for this item, in format YYYY-MM-DD HH:MM:SS
4	Reference_Document	Written by GoldSim to the Run Log (an optional string)
5	Document_ID	Written by GoldSim to the Run Log (an optional string)
6	DTN	Written by GoldSim to the Run Log (an optional string)
7	MOL	Written by GoldSim to the Run Log (an optional string)
8	DOC_Path	Network path for the source document (required only for File elements, as discussed below)
9	DOC_SIG	CRC signature for File source document (a string required only for File elements)
10	Parameter_Note	Text that is imported into the element's Note. This must be a "Memo" field, and does not support rich text or HTML. Only plain text can be imported.

The CRC signature is an alphanumeric code that can be used to uniquely identify whether the file contents have changed. When you download a file, GoldSim compares the CRC signature of the downloaded file with the original signature that was stored in the database. If these are not identical (indicating that the file has been changed), the download will fail.

You can generate a CRC signature for a file using the EFIViewer, a small utility program that is installed with GoldSim.

Value Component Table

The Value Component Table (GS_Value_Component) stores the actual data values. There are one or more records for each data value. Each record must contain the following fields:

No.	Field	Notes
1	Value_ID	The index used to link from the GS_Parameter_Value table
2	Component_ID	Unique index
3	Type_Code	Row number for sorting rows in vectors and matrices (used only arrays).
4 - 63	Value_Column_1, …, Value_Column_60	Data values, to support tables and matrices with up to 60 columns.

For a Data element, the data value is stored in Value_Column_1.

For vector Data elements, the data for each item is stored in Value_Column_1, and there should be one record for each row (item) in the vector. For matrix Data elements, the data for each row of the matrix is stored in Value_Column_1 through Value_Column_nn (where nn is as specified in the Parameter_Code field of the Parameter Table), and there should be one record for each row of the matrix.

For a 1-D Table element, the independent variable values are stored in Value_Column_1, and the dependent variable values are stored in Value_Column_2. There is one record for each row in the table.

For a 2-D Table element, the row independent variable values are stored in Value_Column_1, and the dependent variable values are stored in Value_Column_2 through Value_Column_n, where n is one greater than the number of columns in the table. The first record for a 2-D Table element contains values for the column independent variable in Value_Column_2 through Value_Column_n, and the successive records contain values for the Row independent variable followed by values for the dependent variable.

Note: Matrices and 2-D tables can have no more than 60 columns.

For Stochastic elements other than discrete, cumulative and sampled results, the arguments are stored, in sequence, in the Value_Column_1 … Value_Column_n entries. The order of the arguments for each type of Stochastic is listed below:

Stochastic	Order of Arguments
Uniform	minimum, maximum
Log-uniform	minimum, maximum
Normal	mean, standard deviation
Truncated Normal	mean, standard deviation, minimum, maximum
Log-normal (geometric)	geometric mean, geometric standard deviation
Truncated Log-normal (geometric)	geometric mean, geometric standard deviation, minimum, maximum
Log-normal (true mean)	true mean, true standard deviation
Truncated Log-normal (true mean)	true mean, true standard deviation, minimum, maximum
Triangular	Minimum (or 10%), most likely, maximum (or 90%)
Log-triangular	Minimum (or 10%), most likely, maximum (or 90%)
Poisson	expected value
Beta	Successes, Failures
Generalized Beta	Mean, standard deviation, minimum, maximum
BetaPERT	Minimum (or 10%), most likely, maximum (or 90%)
Gamma	mean, standard deviation
Truncated Gamma	mean, standard deviation, minimum, maximum
Weibull	minimum, slope, mean-mimimum
Truncated Weibull	minimum, slope, mean-minimum, maximum
Binomial	# picks, probability of success
Boolean	probability of true
Student's t	degrees of freedom
Exponential	mean
Pareto	a, b

Stochastic	Order of Arguments
Truncated Pareto	a, b, maximum
Negative Binomial	successes, probability of success
Extreme Value (minimum)	location, scale
Extreme Value (maximum)	location, scale
Extreme Probability (mimimum)	number of samples
Extreme Probability (maximum)	number of samples
Pearson type III	location, scale, shape

For a discrete, cumulative or sampled results Stochastic element type, there are multiple rows in the table, with one row for each value. Value_Column_1 contains the probability values, and Value_Column_2 contains the result values. For a sampled results distribution, there is only one value (the result) and this is placed in Value_Column_2 (Value_Column_1 is ignored).

For a Sampled Results distribution, there are multiple rows in the table, with one row for each value. Value_Column_2 contains the result values. Any probabilities entered in Value_Column_1 are ignored (all results have equal weights).

Example File

The Database.gsm file in the General Examples folder in your GoldSim directory provides an example of how elements can be linked to a Yucca Mountain database. This folder also includes the sample database referenced by this file (Yucca_DB.accdb), created using Microsoft Access 2003. In order to use the GoldSim file, you will need to add the database as data sources to your computer.

Read more: Adding Data Sources to Your Computer (page 973).

Appendix F: Integration Methods and Timestepping Algorithm

> **On two occasions I have been asked [by members of Parliament], 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.**
>
> **Charles Babbage**

Appendix Overview

The elements and links in a GoldSim model represent a system of equations. Except in the simplest cases, these are systems of differential equations, and they are often nonlinear and discontinuous. In general, the systems of equations that GoldSim must solve will not have an analytical solution (i.e., they cannot be solved exactly), and must be solved *numerically* (i.e., using an algorithm that provides a numerical approximation to the actual solution).

In order to effectively use GoldSim, particularly for complex problems, it is important to have a basic understanding of the factors affecting the accuracy of your model, and the nature of the numerical approximations used by GoldSim.

This appendix provides a brief discussion of these numerical algorithms.

In this Appendix

This appendix discusses the following:

- Factors Affecting the Accuracy of Simulation Models
- Primary Numerical Approximations in GoldSim
- Summary of GoldSim's Dynamic Timestepping Algorithm

Factors Affecting the Accuracy of Simulation Models

A simulation model is an abstract representation of an actual (or hypothetical) system. By definition, it is a simplification of reality, with the goals being to include those aspects that are assumed to be important and omit those which are considered to be nonessential, so as to derive useful predictions of system performance in an efficient manner.

In addition, for most real world systems, there will be significant uncertainty regarding the processes that are controlling the system and the parameter values describing those processes. As a result, the most important factor impacting the accuracy of your model is the degree to which your conceptual and mathematical model have captured reality and the degree to which you have quantitatively represented your uncertainty in the system. In most real world systems that you would simulate in GoldSim, *the uncertainty in the simulated result due to your uncertainty in the processes and parameters controlling the system will be far greater than any inaccuracies introduced by the numerical solution method.* As a result, it will generally be more worthwhile for you to spend your time ensuring that your model captures the key aspects of the system realistically rather that worrying about the numerical accuracy of the solution.

Having said that, it is still important to understand the nature of the inaccuracies that can arise from GoldSim's numerical approximations in order to ensure that these do indeed remain small. The primary numerical factors affecting the accuracy of a GoldSim model are as follows:

- **Integrating Differential Equations:** GoldSim solves differential equations by numerically integrating them (via Stocks and Delays). This numerical integration is the largest potential source of numerical inaccuracies in a GoldSim model.
- **Solving Coupled Equations:** In some cases, the system you wish to model may include coupled equations or coupled differential equations. Solutions of these types of equations can be computationally-intensive (e.g., nonlinear coupled differential equations). For some specific types of coupled systems, GoldSim provides fast and accurate solution techniques. In other cases, these equations must be solved approximately using Previous Value elements. The use of Previous Value elements in this case can introduce numerical approximations that are of the same order as those introduced via numerical integration of ordinary differential equations.

  Read more: Using Advanced Algorithms to Solve Coupled Equations (page 1080).
- **Representing Discrete Events:** In many real world systems, discrete events occur which impose discontinuous changes onto the system. Superimposing such discontinuities onto a continuously-varying system discretized in time can introduce inaccuracies. GoldSim provides a powerful timestepping algorithm for accurately representing such systems.

These items are discussed in the topics below.

Note: Round-off error is sometimes noted as an important source of error in simulation models. Although this can indeed be important for some specialized engineering and science simulation tools, given the nature of GoldSim applications, and the fact that GoldSim uses double-precision to carry out its calculations, it is highly unlikely that round-off error could ever have a noticeable impact on a GoldSim model.

Primary Numerical Approximations in GoldSim

GoldSim Numerical Integration Algorithm

Stocks (Integrators and Reservoirs) represent integrals of the form:

$$\text{Value} = \text{Initial Value} + \int (\text{Rate of Change})dt$$

The Rate of Change, of course, can be a function of time.

In this case, we are solving the following differential equation:

$$\frac{d\text{Value}}{dt} = \text{Rate of Change}; \text{Value}_{t=0} = \text{Initial Value}$$

Numerically, GoldSim approximates the integral shown above as a sum:

$$\text{Value}(t_n) = \text{Initial Value} + \sum_{i=1}^{n} \text{Rate of Change}(t_i - \Delta t_i)\,\Delta t_i$$

where:

Δt_i is the timestep length just prior to time t_i (typically this will be constant in the simulation);

Rate of Change($t_i - \Delta t_i$) is the Rate of Change at time=$t_i - \Delta t_i$; and

Value(t_i) is the value at end of timestep i.

Note that the Value at a given time is a function of the Rate of Change at previous timesteps (but is not a function of the Rate of Change at the current time).

This particular integration method is referred to as ***Euler integration***. It is the simplest and most common method for numerically solving such integrals. The key assumption in the method is that the rate remains constant over a timestep. The validity of this assumption is a function of the length of the timestep and the timescale over which the rate is changing. The assumption is reasonable if the timestep is sufficiently small.

To get a feeling for the errors that can be produced by Euler integration, consider the following very simple integral:

$$\text{Value} = \text{Initial Value} + \int -k * (\text{Value})dt$$

where k is a constant. This is the equation for simple (exponential) first-order decay, and the analytical solution is:

$$\text{Value} = \text{Initial Value} * e^{-kt}$$

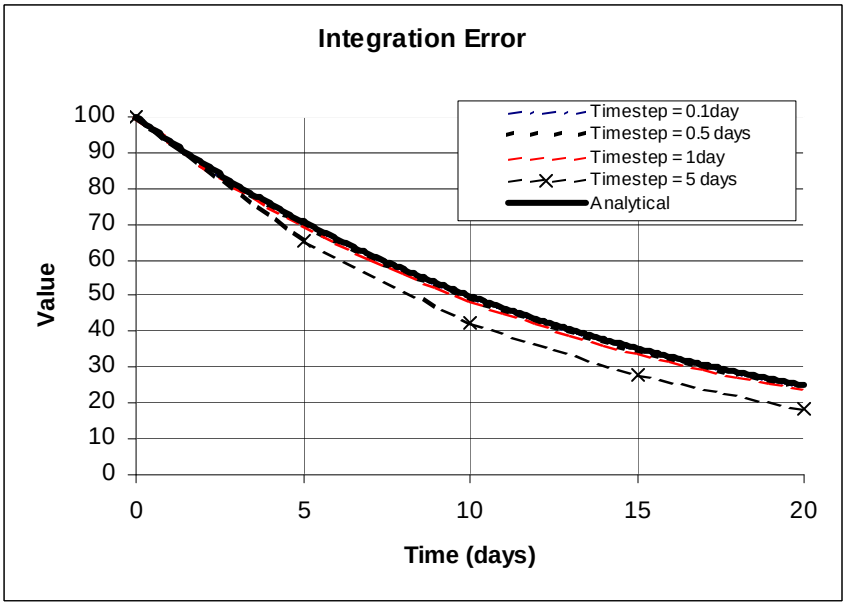
The timescale of the dynamic process can be quantified in terms of a *half-life* (the time it takes the current value to decay to half of its initial value). The half life is equal to 0.693/k. The table below compares the results of computing the

Value analytically and numerically (by comparing the results at 20 days assuming an initial value of 100 and a half-life of 10 days):

Solution	Value at 20 days	% Error*
Analytical Solution	25.00	-
Timestep = 5 days	18.24	9.0%
Timestep = 1 days	23.78	1.6%
Timestep = 0.5 days	24.40	0.8%
Timestep = 0.1 days	24.89	0.1%

*Error computed as $\frac{\text{Simulated} - \text{Analytical}}{\text{Analytical} - \text{Initial Value}}$

A plot of these results is shown below:



As can be seen, with the exception of the 5 day timestep, the Euler integration method is relatively accurate. In fact, for most systems that you will be simulating using GoldSim, a numerical error of several percent is likely to be acceptable (and much smaller than the error caused by the uncertainty in the initial conditions, the parameters, and the conceptual model). As a general rule, your timestep should be 1/3 to 1/10 of the timescale of the fastest process being simulated in your model.

Read more: Selecting the Proper Timestep (page 1078).

Warning: The magnitude of the integration error depends on the nature of the model. In stable models that are dominated by negative feedback loops (and hence tend toward equilibrium), the errors tend to diminish with time. Systems that are unstable and grow exponentially or oscillate with no damping tend to accumulate errors over time. For these types of systems (e.g., a swinging pendulum), a very small timestep may be required to accurately simulate the system using Euler integration.

Note: In cases where a small timestep is required to maintain accuracy, this can be done in a very computational efficient way by using Containers with Internal Clocks, which allow you to locally use a much smaller timestep.

Read more: Specifying Containers with Internal Clocks (page 434).

Other Integration Methods

Although the Euler method is simple and commonly used, other integration methods exist which can achieve higher accuracy with a larger timestep (e.g., Runga-Kutta, variable timestep methods). Because these methods can use a much larger timestep without losing accuracy, they are more computationally efficient.

These methods, while being important for some types of systems (e.g., sustained oscillators like a pendulum), are not incorporated into GoldSim for the following reasons:

- Higher order methods are most useful when simulating physical systems in which the mathematical model, initial conditions and input parameters are known very precisely, and small integration errors can be significant. For the most part, the kinds of systems that you will be simulating using GoldSim can be handled effectively using Euler integration.
- Higher-order methods work best when simulating continuously-varying systems. They do not deal well with systems that behave discontinuously and/or are impacted by discrete changes. Most real world systems do not vary continuously, and GoldSim therefore provides powerful algorithms for accurately superimposing discrete changes (discontinuities) onto a continuously-varying system. These algorithms are incompatible with higher-order integration methods.

  Read more: Accurately Simulating Discrete Events that Occur Between Timesteps (page 1079).
- For some kinds of systems (e.g., mass or heat transport using the Contaminant Transport Module), GoldSim uses powerful algorithms to accurately solve nonlinear coupled differential. These algorithms dynamically adjust the timestep during the simulation, and are incompatible with higher-order integration methods.

  Read more: Using Advanced Algorithms to Solve Coupled Equations (page 1080).

As mentioned above, in cases where a small timestep is required to maintain accuracy using Euler integration, this can be done in a very computational efficient way by using Containers with Internal Clocks, which allow you to locally use a much smaller timestep.

Read more: Specifying Containers with Internal Clocks (page 434).

Approximate Solutions to Coupled Equations

In some situations, you may wish to simulate a static or dynamic processes in which the variables in the system are coupled such that they respond instantaneously to each other, with no time lags. In GoldSim, these are referred to as *recursive loops*, and they are treated differently from feedback loops.

If you encounter a system such as this in one of your models, you can handle it in one of two ways. First, you could solve the system of equations directly either prior to running the model (e.g., using substitution), or dynamically while running the model (e.g., using an External element or GoldSim's matrix

functions to solve the appropriate equations). Note, however, that for many complex models (e.g., non-linear coupled equations), the solution could be very computationally intensive.

GoldSim offers an alternative: solve the equations approximately and iteratively using ***Previous Value elements***. A Previous Value element allows you to reference the previous value (i.e., the previous timestep) of an output.

Read more: Creating Recursive Loops Using Previous Value Elements (page 901).

Selecting the Proper Timestep

As a general rule, your timestep should be 3 to 10 times shorter than the timescale of the fastest process being simulated in your model. In simple systems, you should be able to determine the timescales of the processes involved. In more complex models, however, it may be difficult to determine the timescales of all the processes involved.

In addition, for some kinds of models (e.g., some oscillating systems), this general rule may not be sufficient and you may need a smaller timestep).

Therefore, after building your model, you should carry out the following experiment:

1. Carry out an expected value or median value simulation.
2. Reduce the timestep length by half (increase the number of timesteps by a factor of 2), rerun the model, and compare results.
3. Continue to half the timestep length until the differences between successive simulations are acceptable.

Summary of GoldSim's Dynamic Timestepping Algorithm

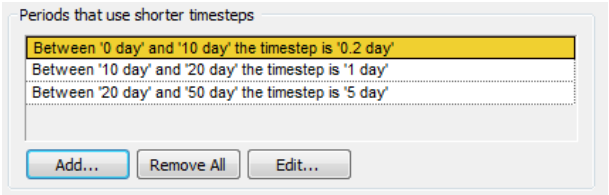
GoldSim provides a powerful timestepping algorithm that can dynamically adjust to more accurately represent discrete events, respond to rapidly changing variables in your model, represent specified SubSystems in your model using different timesteps, and accurately represent some special kinds of systems (e.g., mass and heat transport within the Contaminant Transport Module).

These features are discussed in detail elsewhere in this document. To complement the rest of the information provided in this appendix, however, these features are summarized here.

Defining Specific Periods with Shorter Timesteps

GoldSim allows you to increase or decrease the timestep length according to a specified schedule during a simulation (e.g., start with a small timestep, and then telescope out to a larger timestep). This can be useful, for example, if you know that early in a simulation, parameters are changing rapidly, and hence you need a smaller timestep.

You do this by defining Periods Steps, which have different durations and timestep lengths. An example of pre-specified time periods is shown below:



In this example, the model uses a 0.2day timestep for the first 10 days, a 1 day timestep between 10 days and 20 days, and a 5 day timestep between 20 days and 50 days

(Although not indicated here, it then reverts to the default timestep of 10 days for the remainder of the simulation).

Read more: Adding Shorter Timesteps Over Defined Periods (page 426).

Dynamically Adjusting the Timestep

Although defining shorter timesteps over defined periods can be very useful, you must fully specify them prior to running the simulation. That is, you must know how you would like to alter your timestep prior to running the model. In some cases, however, it may not be possible to do this. That is, in complex systems (particularly ones with uncertain parameters), variables may change at different rates in different realizations, in ways that you cannot predict prior to running the model.

To better simulate these kinds of systems, GoldSim provides an advanced feature that allows you to dynamically adjust the timestep during a simulation (i.e., insert "internal" timesteps) based on the values of specified parameters in your model. For example, you could instruct GoldSim to use a timestep of 1 day if X was greater than Y, and 10 days if X was less than or equal to Y. Similarly, you could instruct GoldSim to use a short timestep for a period of 10 days after a particular event occurs, and then return to the default timestep.

Read more: Dynamically Controlling the Timestep (page 431).

Assigning Different Timesteps to SubSystems

In addition to providing a dynamic timestepping algorithm on a global scale (i.e., for the entire model), GoldSim also enables you to apply dynamic timestepping to specific Containers. This allows you to specify different timesteps for different parts (i.e., Containers) in your model. For example, if one part of your model represented dynamics that changed very rapidly (requiring a 1 day timestep), while the rest of the model represented dynamics that changed much more slowly (requiring a 10 day timestep), you could assign a 10 day timestep to the model globally, and a 1 day timestep to the container representing the SubSystem that changed rapidly.

Read more: Specifying Containers with Internal Clocks (page 434).

Accurately Simulating Discrete Events that Occur Between Timesteps

In some cases, events or other changes in the model may not fall exactly on a scheduled update. That is, some events or changes may actually occur between scheduled updates of the model. These trigger an "unscheduled update" of the model. Unscheduled updates are timesteps that are dynamically inserted by GoldSim during the simulation in order to more accurately simulate the system. That is, they are not specified directly prior to running the model. GoldSim inserts them automatically (and, generally, without you needing to be aware of it).

"Unscheduled updates" can be generated in the following ways:

- When events are ouput by a Timed Event, Event Delay, Discrete Change Delay or Time Series element;
- By manually specifying a dynamic timestep (i.e., dynamically controlling the time between updates);
- When a Reservoir element reaches an upper or lower bound;
- When a Resource becomes exhausted;
- When any element is triggered by an *At Stock Test*, *At Date* or *At Etime* triggering event; and
- By some specialized elements in GoldSim extension modules (Action and Function elements in the Reliability Module, Fund elements in the

Financial Module, and Cell elements in the Flow Module and Contaminant Transport Module).

When any of these events occur, GoldSim automatically inserts an unscheduled update at the exact time that the event or change occurs. For example, if you had specified a one day timestep, and aTimed Event occurs at 33.65 days (i.e., between the scheduled one-day updates), GoldSim would insert an unscheduled update at 33.65 days.

Read more: Understanding Timestepping in GoldSim (page 415).

By default, scheduled updates are always dynamically inserted by GoldSim. However, in some (rare) cases, you may want to prevent unscheduled updates from being inserted. For example, if your model included a specialized algorithm that was designed based on the assumption that the timestep was constant, inserting unscheduled updates could invalidate the algorithm. To support such situations, GoldSim allows you to disable unscheduled updates.

Warning: Because unscheduled updates are intended to more accurately represent a complex dynamic system, disabling this feature should be done with caution, and is generally not recommended.

Read more: Controlling Unscheduled Updates (page 430).

Using Advanced Algorithms to Solve Coupled Equations

For some special types of systems, GoldSim provides additional dynamic timestepping algorithms (different from the timestep algorithms described above) to more accurately solve these equations. For example, the Contaminant Transport Module utilizes dynamic timestep adjustment to solve the coupled differential equations associated with mass and heat transport.

This algorithm allows GoldSim to handle "stiff" systems (systems with widely varying time constants) as well as nonlinear aspects of the system in a very accurate and computationally efficient manner. This algorithm is discussed in the **GoldSim Contaminant Transport Module User's Guide**.

Glossary of Terms

Active Scenario

When scenarios have been defined, the scenario that is being viewed when you are browsing a model.

Affects View

A special browser view in GoldSim that allows you to see all the elements the selected element affects.

Alias

The name by which an exposed output of a localized Container is referenced.

Allocator

An element that allocates an incoming signal to a number of outputs according to a specified set of demands and priorities. Typically, the signal will be a flow of material (e.g., water), but it could also be a resource, or a discrete transaction.

Array

A collection of variables that share common output attributes and can be manipulated in GoldSim elements or input expressions.

Array Labels

A collection of labels identifying the items of an array.

Autocorrelate

To correlate a Stochastic element to a previous value of itself.

Browser

An alternative view of a GoldSim model, in which elements are displayed in a tree, and organized either hierarchically, or by type.

Built-in Constants

Constants (such as pi) that are built-in to GoldSim and can be used when creating expressions in input fields.

Built-in Functions

Mathematical functions (such as sine, maximum, round) that are built-in to GoldSim and can be used when creating expressions in input fields.

Cancellation Code

An alphanumeric code generated by GoldSim that can be used to verify that a registration has been terminated.

Causality Sequence

The specific order in which GoldSim updates (computes) elements every timestep.

Change Note

A note added to an element when using versioning that is subsequently displayed when showing changes.

Chart Style

A collection of settings for a particular type of result display chart.

Clones

Sets of elements whose properties change simultaneously when any one member of the set is edited.

Complementary Cumulative Distribution Function

The complement of the cumulative distribution function.

Conceptual Model

A representation of the significant features, events and processes controlling the behavior of a system.

Condition

An output Type. A Condition is a Boolean switch (e.g., Yes/No, True/False, On/Off).

Conditional Expression

An expression which evaluates to (produces) a Condition (rather than a Value).

Conditional Tail Expectation

The expected value of the output given that it lies above a specified quantile. That is, it represents the mean of the worst $100(1 - \alpha)\%$ of outcomes, where α is the specified quantile.

Container

An element that acts like a "box" or a "folder" into which other elements can be placed. It can be used to create hierarchical models.

Convolution Element

An element that solves a convolution integral.

Coupled Link

A link between two elements which causes the elements to be solved in a coupled (rather than a sequential) manner. Coupled links can only be created when using an extension module.

Cumulative Distribution Function

The integral of a probability density function.

Data Source

A source of data external to your GoldSim model that can be automatically imported into GoldSim elements. External data sources are either spreadsheets, text files, databases or DLLs.

Date-time Simulation

A simulation that tracks time using the simulated Date/time.

Delay Elements

A class of elements that simulate processes that delay continuous or discrete signals and flows. The output of a delay element lags its inputs.

Deterministic Simulation

A simulation in which the input parameters are represented using single values (i.e., they are "determined" or assumed to be known with certainty).

Dimensions

An output attribute for an element that defines the dimensionality (in terms of Length, Time and other fundamental dimensions) of the output.

Discrete Change

An element that generates discrete change signals that can subsequently modify stock elements.

Discrete Change Signal

A discrete signal that contains information regarding the response to an event.

Discrete Event Signal

A discrete signal indicating that something (e.g., an accident, an earthquake, a bank deposit) has occurred.

Discrete Signal

A special category of output that emit information discretely, rather than continuously.

Display Units

The units (e.g., m, g, $/day) in which an output is displayed within GoldSim.

Drawing Tools Toolbar

A toolbar at the top of the GoldSim interface that provides buttons for adding graphical components to your model.

Edit Mode

The state of a model when it is being edited and does not contain simulation results.

Elapsed Time Simulation

A simulation that tracks time using the elapsed time.

Euler Integration

A simple and commonly used numerical integration method.

Exposed

Description of an output within a localized Container that can be referenced outside of the Container.

Expression Element

A function element that produces a single output by calculating user-specified mathematical expressions.

External Functions

User-defined modules that can be linked to GoldSim at runtime as Dynamic Link Libraries (DLLs).

Feedback Loop

A circular system in which the variables in the loop impact each other, but do not respond instantaneously. Feedback loops contain at least one state variable.

Function Elements

Elements that instantaneously compute outputs based on one or more defined inputs.

Function Of View

A special browser view in GoldSim that allows you to see all the elements that affect the selected element.

Graphical Objects

Objects in GoldSim that are used to embellish or document the model.

Graphics Pane

The primary portion of the GoldSim interface, where the graphical depiction of the model is shown.

History Generator

An element that generates stochastic time histories of variables. A stochastic time history is a random time history that is generated according to a specified set of statistics.

Importance Sampling

An algorithm that biases sampling of probability distributions in order to better resolve the tails of the distributions.

Information Delay

A delay element that delays information signals, and does not enforce conservation of the signal.

Input Elements

Elements that are used to define basic inputs for a model.

Installation Code

An alphanumeric code which is automatically generated when you install GoldSim.

Interactive Result

A result display that is shown in a modal window (i.e., windows that always retain the focus). Interactive results can be converted to modeless Result elements.

Keywords

Text delimited by the % symbol (e.g., %x_unit%) that can be used when creating chart styles to insert context sensitive text (e.g., for axis labels).

Kurtosis

A measure of how "fat" a distribution is relative to a normal distribution with the same standard deviation. A normal distribution has a kurtosis of 0. The kurtosis is a function of the fourth moment of the distribution.

Latin Hypercube Sampling

A stratified sampling method that has the effect of better ensuring that the space of the parameter is uniformly spanned.

Link Cursor

A special cursor for creating links invoked by double-clicking on an input or output object in a browser.

Live Model

When using GoldSim's scenario features, a "scratch" model, or a temporary placeholder model where you can experiment before saving something as a scenario.

Localize

An action that you can apply to a Container that creates a separate scope for the elements in that Container.

Lookup Table Element

A function element that allows you to create a 1, 2, or 3-dimensional lookup table (response surface).

Material

Tangible things (water, dirt, cash, widgets) that are tracked in a simulation.

Material Delay

A delay element that delays flows of materials (e.g., masses, volumes, items).

Mathematical Model

A set of input assumptions, equations and algorithms describing the behavior of a system.

Matrix

A two-dimensional array.

Mean

The expected value (average) of a distribution. It is the first moment of the distribution.

Median

The 50th percentile of a distribution.

Menu Bar

A bar at the top of the GoldSim interface that provides access to menus from which nearly any GoldSim operation can be carried out.

Model Objects

Objects in GoldSim that are used to quantitatively represent the variables and relationships in a model. The primary model object is the element.

Model Root

The top-level Container in a GoldSim model.

Modes

The states that a GoldSim model can be in at a given time. There are three modes: Edit Mode, Run Mode, and Result Mode.

Monte Carlo Simulation

A method for propagating (translating) uncertainties in model inputs into uncertainties in model results.

Normal Link

A standard link between two elements.

Note Pane

A dockable window in GoldSim that displays a Note associated with the selected element.

Numerical Integration

An approximate solution to an integral equation, carried out by discretizing a variable (e.g., time) into discrete intervals.

Output Attributes

Three properties of an element's output that determine the kinds of inputs to which it can be linked: type, order and dimensions.

Pan Cursor

A special cursor for panning the graphics pane invoked via the Zoom toolbar.

Performance Measure

A specific model output by which you judge the performance of a system.

Plot Points

Points in time at which the value of outputs are saved for plotting time history data.

Ports

Small arrows on the side of the element in the graphics pane. You can left-click on a port to access the element's inputs and outputs.

Precedence

The order in which mathematical operators are evaluated in an expression.

Previous Value Element

An element that outputs the value of its input from the previous model update.

Primary Output

For an element with multiple outputs, the output that has the same name as the element.

Probabilistic Simulation

A simulation in which the uncertainty in input parameters is explicitly represented by defining them as probability distributions.

Probability Density Function

A plot of the relative likelihood of the values of an uncertain variable.

Probability Distribution

A mathematical representation of the relative likelihood of a variable having certain specific values.

Probability Histories

A probabilistic representation of the time history of an output in which the percentiles for multiple realizations are plotted.

Probability Mass Function

A plot of the relative likelihood of the values of a discrete uncertain variable.

Realization

A single model run within a Monte Carlo simulation. It represents one possible path the system could follow through time.

Recursive Loop

A system with circular logic that does not contain any state variables.

Reference Version

The version to which your current model is compared when tracking changes.

Registration Code

An alphanumeric code provided to you when you license GoldSim. This code is required in order for you to register and run GoldSim.

Registration Key

An alphanumeric code sent to you if you are manually registering GoldSim (rather than registering it via the Internet).

Reporting Periods

Regular time points during a simulation (e.g., every month, every year) at which you can compute and view results associated with that period (e.g., monthly averages, annual cumulative values).

Reservoir

A stock element that integrates and conserves flows of materials.

Resource

Something that has a limited supply (e.g., spare parts, fuel, skilled personnel, money) and is required in order for elements of the modeled system to carry out certain actions.

Result Element

An element that can be used to organize, analyze and display results.

Result Mode

The state of a model when it has been run and contains simulation results for a single set of input parameters.

Run Controller

A dialog used to control the manner in which a GoldSim simulation is run.

Run Log

Text that is stored with a GoldSim model that is in Result Mode. It contains basic information regarding the simulation, and any warning or error messages that were generated.

Run Mode

The state of a model when it is running.

Run Properties

A set of fundamental properties that track the progress of the simulation (e.g., Time, Realization) and can be referenced like outputs in expressions.

Scalar

An output consisting of a single value or condition.

Scenario Data

Data elements that differentiate the various scenarios in a model.

Scenario Manager

A dialog that allows you to create, define and run scenarios.

Scenario Mode

The state of a model when it contains scenario results, allowing multiple scenarios to be compared.

Scenarios

Models with specific sets of input parameters. In particular, different scenarios have different values for one or more Data elements.

Scheduled Timesteps

Timesteps that are directly specified by the user prior to running the model.

Scope

The portion of a model from which an element's output can be referenced. You cannot reference an element in a different scope unless that output is specifically exposed.

Script Element

An element that can be used to create a list of executable statements (e.g., variable definitions, variable assignments and statements controlling the sequence of execution such as loops and if statements) in order to implement complex logic and operations.

Secondary Output

For an element with multiple outputs, an output that has a different name than the element.

Simulation Model

The implementation of a mathematical model of a system within a specific computational tool (or set of tools).

Skewness

A measure of the symmetry of a distribution. A symmetric distribution has a skewness of 0. The skewness is a function of the third moment of the distribution.

Splitter

An element that splits an incoming signal between a number of outputs based on specified fractions or amounts. Typically, the signal will be a flow of material (e.g., water), but it could also be a resource, or a discrete transaction.

Spreadsheet Element

An element that can dynamically link to an Excel spreadsheet.

Standard Deviation

The square root of the variance of a distribution. The variance is the second moment of the distribution and reflects the amount of spread or dispersion in the distribution.

Standard Toolbar

A toolbar at the top of the GoldSim interface that provides buttons for common GoldSim actions.

State Variable

The output of an element in GoldSim whose value is computed based on the historical value of the element's inputs (as opposed to only being a function of the current value of the element's inputs). State variables have well-defined initial conditions. Feedback loops can only be created if they contain at least one state variable.

Status Bar

A bar at the bottom of the GoldSim interface that provides information regarding the status of the model.

Stochastic

An element that can be used to quantitatively represent the uncertainty in a model input.

Stock Elements

A class of elements that numerically integrate inputs, and hence are responsible for internally generating the dynamic behavior of many systems.

Store

Stockpiles or places where a Resource (e.g., parts, personnel) is stored or located when not being used. Resource Stores can be thought of as having physical locations in the system you are modeling. The can be global or local (associated with a Container).

Style File

An external file which stores chart style that you can import.

SubModel

A specialized element that allows you embed one complete GoldSim model within another GoldSim model. This facilitates, among other things, probabilistic optimization, explicit separation of uncertainty from variability, and manipulation of Monte Carlo statistics.

SubSystem

A specialized Container that is completely "self-contained". SubSystems can take on some useful features and properties (e.g., conditionality, having an internal clock, and being able to loop), but also have some limitations (with regard to how they can be incorporated into feedback loops).

System Units File

A file on your hard drive (named units.dat) that stores all of your unit settings.

Table Function

A special function for referencing user-defined lookup tables that can be referenced in input fields. It is automatically created whenever you create a Lookup Table element.

Timed Event

An element that generates discrete event signals based on a specified rate of occurrence.

Timestep

A discrete interval of time used in dynamic simulations.

Total System Model

A simulation model that focuses on creating a consistent framework in which all aspects of the system, as well as the complex interactions and interdependencies between subsystems, can be represented.

Unit Categories

A category of units defined by a name (e.g., Area) and a specific set of dimensions (Length^2).

Unit Strings

Strings containing only unit abbreviations used to define compound units (e.g., m/sec, lbf/m2).

Unscheduled Updates

Timesteps that are inserted automatically by GoldSim during a simulation and are not directly specified by the user prior to running the model.

Vector

A one-dimensional array.

Version

A "snapshot" of your model at a particular point in time.

Versioning

The process of tracking changes that you make to your model file.

Index

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z