CONTINUOUS PROCESS MODELING

QUICK START GUIDE



extendsim

ANDRITZ

© 2026 ANDRITZ Inc. This program is protected by US and international copyright laws.

You may not copy, transmit, or translate all or any part of this document in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than your personal use without the prior and express written permission of ANDRITZ Inc.

License, Software Copyright, Trademark, and Other Information

The software described in this manual is furnished under a separate license and warranty agreement. The software may be used or copied only in accordance with the terms of that agreement. Please note the following:

> ExtendSim blocks and components (including but not limited to icons, dialogs, and block code) are copyright © by ANDRITZ Inc. and/or its Licensors. ExtendSim blocks and components contain proprietary and/or trademark information. If you build blocks, and you use all or any portion of the blocks from the ExtendSim libraries in your blocks, or you include those ExtendSim blocks (or any of the code from those blocks) in your libraries, your right to sell, give away, or otherwise distribute your blocks and libraries is limited. In that case, you may only sell, give, or distribute such a block or library if the recipient has a valid license for the ExtendSim product from which you have derived your block(s) or block code. For more information, contact ANDRITZ at Info.ExtendSim@Andritz.com or Support.ExtendSim@Andritz.com.

© 2026 ANDRITZ Inc. This program is protected by US and international copyright laws. Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation. All other product names used in this manual are the trademarks of their respective owners. All other ExtendSim products and portions of products are copyright by ANDRITZ Inc. All right, title and interest, including, without limitation, all copyrights in the Software shall at all times remain the property of ANDRITZ Inc. or its Licensors.

Acknowledgments

Extend was created in 1987 by Bob Diamond; it was re-branded as ExtendSim in 2007.

The contents of this document are the result of years of work by software architects, simulation engineers, and technical writers and editors of ExtendSim products.

ANDRITZ Inc • 13560 Morris Road, Suite 1250 • Alpharetta, GA 30004 USA
408.365.0305 • Info.ExtendSim@Andritz.com
www.ExtendSim.com

Table of Contents

Continuous Process Quick Start **1**

Introduction **1**
- Welcome! 1
- About the Quick Start Guides 1
- Who should read this Quick Start Guide 1
- Chapters in this Quick Start Guide 1
- What is CP modeling? 2
- Why simulate using CP modeling? 2
- Where is continuous process modeling being used? 2
- How ExtendSim CP is being used 3
- How CP modeling compares to other methodologies 7
- Where to get more information 7

Model Basics **9**
- Overview 9
- Opening and exploring a model 9
- Model basics 12
- Blocks 13
- Connectors and connections 15
- Libraries 16
- Blocks for building continuous models 17

Tutorial **21**
- Overview 21
- How to use ExtendSim CP 21
- A simple continuous process system 21
- Launch ExtendSim 22
- Set the simulation parameters 22
- Blocks used for the simple Reservoir model 23
- Placing and connecting blocks on the model worksheet 23
- Make connections and adjust the blocks 26
- Enter dialog parameters and settings 26
- Configure the Line Chart block 28
- Your tutorial model 30
- Running a simulation 30
- Verify and validate results 31
- Save the model 31
- Additional models 31
- What's next 32

Concepts & Terminology **35**
- Simulation timing 35
- Delta time and step size 35
- Feedback and delays 37
- Integration 38
- Simulation order 40
- Mixing block types 41
- Connections to multiple inputs 41

The Big Picture **43**

Overview....43
Using equation-based blocks....43
Using the Python Bridge block....44
Creating custom blocks....44
Model-building to represent the dynamics of the system....45
Creating a user interface for the model....45
Getting data into the model....46
Managing data for use in the model....47
Analyzing data and assessing results....48
Reporting results and exporting data....49
Communicating with external applications and devices....51

Continuous Process Quick Start

Introduction

Welcome!

Welcome to ExtendSim — an advanced platform for building and analyzing simulation models, with powerful tools for insight and decision-making.

About the Quick Start Guides

The purpose of a Quick Start Guide is to get new users quickly familiar with a specific ExtendSim module and aware of the ExtendSim features and capabilities. There are three Quick Start Guides—Continuous Process Modeling, Discrete Event Simulation, and Discrete Rate Modeling. Depending on the product purchased, one or more of these will be available within ExtendSim via the Help menu, in the installed Documentation folder, and online at the ExtendSim website for the latest updates, examples, and videos.

✦ Each of the Quick Start Guides has similar structure and information, so it is not necessary to read all of them.

For more in-depth information, this guide refers to the User Reference and other documentation available in the Help menu and Documents/ExtendSim/Documentation folder.

Who should read this Quick Start Guide

This guide is for model developers who create *continuous process models*. These models simulate continuous systems where time advances in equal steps and state variables, which describe the system at any point in time, change continuously as time advances.

Chapters in this Quick Start Guide

1) Introduction to Continuous Process (CP) modeling:
    * What it is
    * Where and when to use CP modeling
    * How to use it and how it is being used
    * How CP modeling compares to discrete event and discrete rate simulation
    * Where to get more information
2) Model basics and how to run a model
3) Tutorial: building a small continuous process model
4) Concepts and terminology specific to continuous process modeling
5) The Big Picture: features and capabilities you'll want to know about when building models

What is CP modeling?

Continuous process (CP) modeling is the creation of a computer model of a system or process that changes continuously with respect to time. In a CP model:

- Time advances in equal steps
- State variables, which describe the system at any point in time, change continuously as time advances

The model of the system or process may consist of algebraic, differential, or difference equations set up to change continuously with time. During the simulation run, system behavior is tracked by observing the state of the system over the sequence of time steps.

✦ There are three main simulation methodologies: continuous process, discrete event, and discrete rate. For information about the other simulation methods, see "How CP modeling compares to other methodologies" on page 7.

⚠ While continuous process modeling is sometimes equated with System Dynamics, CP modeling is a much broader methodology. System Dynamics is a subset of continuous modeling and is most often used when designing or improving policies and strategies. As seen below, CP modeling has a broader range of application areas.

Why simulate using CP modeling?

Discrete event simulation is the methodology most commonly taught in schools. However, in a whole range of situations, systems are not subject to discrete changes in state. Instead the state of the system changes continuously through time.

Continuous systems are widely seen in engineering, economics, and biology and CP modeling is most often used to:

- Perfect processes and products
- Plan resource capacity by determining the rate at which products are being processed or sold
- Test various schedules to maximize process efficiency
- Create simulators of plants and processes for training purposes
- Evaluate system reliability
- Fine-tune distribution control strategies
- Study system constraints and the impact of rate changes
- Analyze the effect of processes on the internal and external environment

Where is continuous process modeling being used?

Computer simulation is indispensable for understanding, analyzing, and predicting the behavior of complex and large-scale systems. It is used to gain an understanding of the functioning of existing systems and to help design new systems by predicting their behavior before they are actually built. The following table gives some of the most common areas where continuous process modeling is used.

Discipline	Fields	Applications
Science	Biology, Biotech, Chemistry, Ecology, Genetics, Mathematics, Medicine, Pharmaceuticals, Physics	Chemical reaction kinetics, thermodynamics, paper making, population dynamics, growth/decay, competition/cooperation, chaos, genetic algorithms
Engineering	Aerospace, Agricultural, Automotive, Control Systems, Electronic, Environmental, Forestry, Material Science, Mechanical, Mining, Nuclear, Petroleum	Hardware design verification, electro-mechanical systems, neural networks, adaptive systems, algorithm validation, signal processing
Business	Finance, Information Technology, Inventory Management, Human Resources, Operations	Forecasting, credit risk analysis, asset pricing, derivatives trading, data flow and sharing, inventory replacement strategies, resource allocation, process flow
Social Sciences	Economics, Gender Studies, Migration, Psychology, Social Dynamics, Urban Studies	Econometric studies, trade policies, relationship modeling, finite capacity planning, economic booms and recessions, immigration policies

✦ If the system or process is event-based, rather than time based, see the Discrete Event or Discrete Rate Quick Start Guide.

How ExtendSim CP is being used

It is most common for continuous process modelers to use the ExtendSim integrated development environment (IDE) to create their own libraries of blocks. Some applications are:

1) The global leading supplier of plants, equipment, and services for hydropower stations, the pulp and paper industry, and metalworking and steel industries uses OEM'd ExtendSim for control logic verification and steady-state and dynamic process modeling in the global kraft pulp industry and for oil sands operations, potash operations, and hard rock mining.

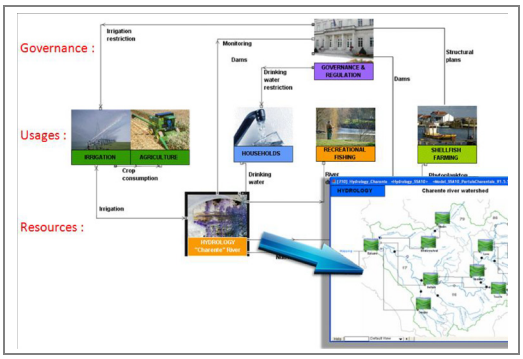
2) The leading consumer products company has created custom ExtendSim blocks to help design machines that process continuous sheet web products such as paper towels, toilet paper, and diapers.

3) A multi-national consulting company uses custom ExtendSim libraries to support industrial and public sector decisions at every point in the water cycle—treatment, distribution, and storage. The goal is to develop projects that provide safe, reliable water supplies. The models help the company and its clients balance economic, social, and environmental needs.



4) A Swedish consulting company uses custom ExtendSim libraries to simulate complex continuous processes in the pulp and paper industry. Models include bleaching operations, fractionation of fiber compositions, and Canadian Standard Freeness (CSF) calculations. Services provided include process optimization, energy savings, cost efficiency, business control, and production control.

5) An EU consortium of 54 partners from 21 countries developed a self-evolving, operational research approach framework for the assessment of policy options for the sustainable management of coastal zone systems. The framework coupled a systems approach with a science-policy interface to improve and enhance deliberations over coastal issues. Models describe a variety of coastal processes ranging from eutrophication, fisheries, and beach attractiveness to clam culture. The simulations contribute to the understanding of social interactions within coastal zone systems and have a positive influence towards understanding the sustainability transition.



6) A USA consulting company creates custom ExtendSim libraries of blocks to model electrical networks and solve time and spatially dependent equations for the conservation of mass, momentum, and energy. They have also developed ExtendSim blocks used to build training simulators for the power (conventional and renewable) and oil & gas (LNG and petrochemical) industries. Plant operators use the simulators to study a plant's dynamics and its transient behaviors. The simulators allow the operators to test different operational procedures and familiarize themselves with process anomalies, helping to minimize operational errors and production losses.

7) A New Zealand pulp and paper mill models their integrated pulp mill, recycle facility, and sack kraft machine to determine the optimum mix of products and grades for specific economic conditions. This mill now boasts bleached softwood kraft production costs in the bottom quartile worldwide.

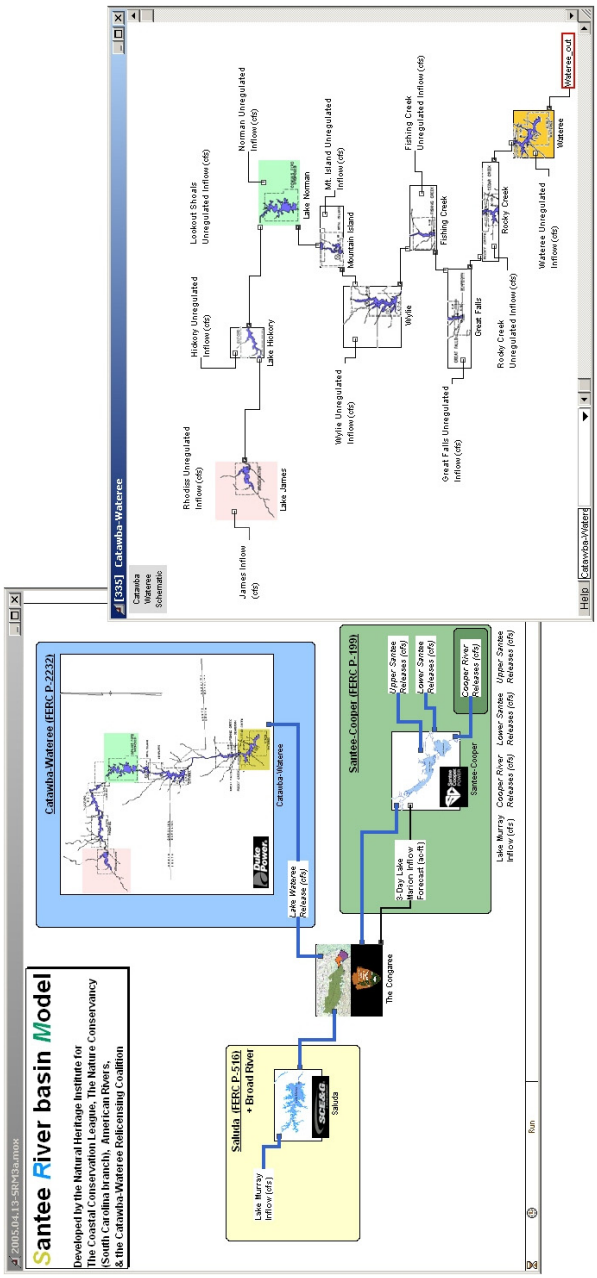


8) A public consortium of scientists and educators developed a curriculum supporting biology education that includes modules that use ExtendSim blocks. For example, a module on environmental decision making allows students to simulate ecosys-

tems and their social and economic forces, while studying the ecosystem's function and productivity.

9) See the next page for a screen shot of a continuous process hydrological model.

Continuous model of the Santee River Basin



Santee River Basin Model (SRM), developed by the National Heritage Institute, et al
This continuous hydrologic model predicts how operational alternatives will affect hydropower generation, lake levels, in-stream flow, and water uses. It analyzes how dam operations can be altered to improve stream flow in over 300 river miles and helps assess the effects on lake levels and power generation. A series of hierarchical blocks with nested submodels keep details in layers below the model's realistic top level. For instance, the Catawba-Wateree hierarchical block contains 9 additional hierarchical blocks, some of which contain even more layers.

How CP modeling compares to other methodologies

The most important aspects of the three main modeling methodologies—continuous, discrete event, and discrete rate—are shown below.

	System State Changes When	Models	Attributes for Tracking	Queued Order Method
Continuous Process	Time changes	Generic stuff	No	N/A
Discrete event	Event occurs	Identifiable things	Yes	FIFO, LIFO, Priority, or a custom order based on Item Attributes
Discrete rate	Event occurs	Generic stuff	Yes	FIFO, LIFO, or a custom order based on Flow Level Attributes

- State change. Models are either event-based or time-based. Continuous process models have a fixed time step where time advances in equal increments and the model changes state at each time step. Discrete event and discrete rate models only change state when an event occurs.
- What is modeled. Continuous and discrete rate models both model homogeneous "stuff", while discrete event models track individually identifiable "things"—entities or items.
- Tracking. Values in continuous models do not have attributes—properties that allow a value to be individually identified and manipulated. However, quantities of flow in discrete rate simulations can be assigned attributes, as can the entities in discrete event models.
- Order method. The order in which the stuff (flow) in discrete rate models or the things in discrete event models leave a queue which is holding more than one quantity or thing. The stuff (values) in continuous models cannot be put into any order.

✦ For more detailed information, see the User Reference chapter titled Modeling Methodologies.

Where to get more information

The ExtendSim documentation, example models, and the video files and documents on the www.ExtendSim.com website provide comprehensive help.

Tutorial & Reference documents

In addition to the Quick Start Guides, there are three Tutorial & Reference documents that are included as eBooks in the Documents/ExtendSim/Documentation folder:

- *ExtendSim Database*. This internal relational database provides model developers with a systematic way to manage information for the model and makes models scalable.
- *Advanced Resource Management*. ARM is a sophisticated architecture for systematically dealing with multiple types of resources and a model's complex resource requirements.
- *Reliability*. Graphically capture and validate complex availability behavior. Determine when scheduled and unscheduled downs occur for individual resources and what impact that has on the availability of the system as a whole.

User Reference

The ExtendSim User Reference has a lot of information you will find helpful when building, using, and presenting models.

How To chapters cover general modeling and simulation topics

- Using libraries and blocks
- Performing analysis
- Enhancing presentations
- Creating a user interface
- Using equation-based blocks
- And much more

Appendices list menu commands and the ExtendSim libraries and blocks

Every menu command is explained; the main libraries are described block by block.

Technical Reference

While it's possible to build continuous models using just the blocks supplied with ExtendSim, it is more likely for continuous modelers to want components with custom behavior. The ExtendSim *integrated development environment (IDE)*—the API, source code editor, compiler, dialog editor, graphical UI builder—makes it easy to modify blocks, develop custom features, and create new blocks with custom code, dialogs, and icons.

As discussed on page 44, ExtendSim also includes sophisticated programming tools such as include files, a code editor, conditional compilation, a source code debugger, code completion, and external source code capabilities.

And very importantly, your custom-built blocks are fully integrated and can be saved in ExtendSim libraries for reuse in other models.

✦ The eBooks ship with the appropriate ExtendSim product. To access these documents, see the Documents/ExtendSim/Documentation/folder or launch the books from the Getting Started model that opens when ExtendSim launches. The User Reference and Technical Reference are also available if you select the Help menu when using ExtendSim.

Example models and videos show you how

ExtendSim includes numerous tutorial models as well as videos and example models that explain concepts discussed in the User Reference. For example models, see the Documents/ExtendSim/Examples folder. For videos, see the ExtendSim website.

Continuous Process Quick Start

Model Basics

Overview

This chapter shows how to open and run a model of a simple continuous system and discusses the components of a model.

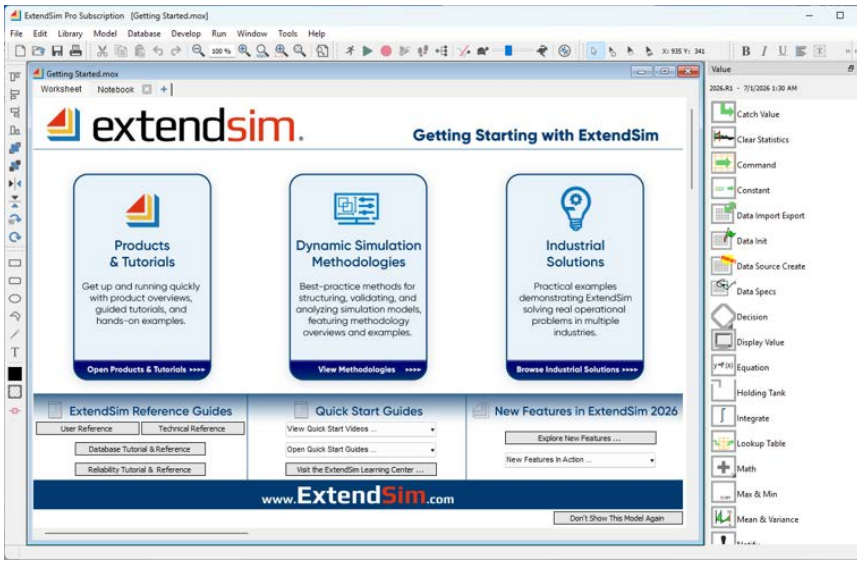
⚠ If you already know how to open, run, and save an ExtendSim model and are familiar with blocks, connectors, and model layout, skip this chapter and go to the next one.

Opening and exploring a model

▸ Launch ExtendSim

As seen on the right, when ExtendSim launches it opens an application window with:

- A model (on the left)
- Library windows (stacked on the right) for every open library
- Toolbars on the top and sides



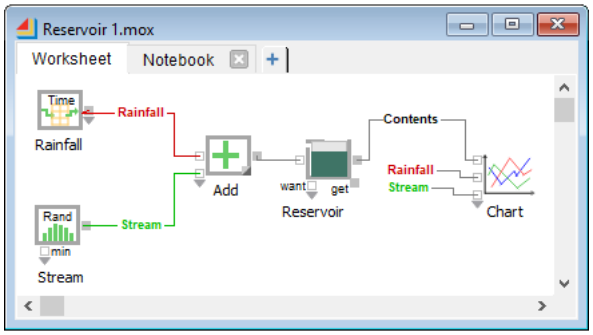
By default, the model that opens will be the Getting Started model. If that option has been removed from the Edit > Options > Model tab, a new blank model will open instead. The Getting Started model has information about ExtendSim products as well as links to the manuals, video tutorials, and more.

This tutorial will focus on the model window. Other aspects of the application window are discussed later in this Guide or in the User Reference.

Open the example model

- Select File > Open
- Browse to Documents/ExtendSim/Examples/Tutorials/Continuous
- Select the model named Reservoir 1
- Click Open

This opens the model window as shown on the right.



About the model

This is a simple example of a reservoir being filled by two water sources—rainfall and an incoming stream. The model uses five blocks (labeled Rainfall, Stream, Add, Reservoir, and Chart) to simulate the continuous system. The purpose of the model is to see how much water accumulates in the reservoir over time.

The model window

Notice that there are two tabs on the top of the model window:

- The *Worksheet* is where the model resides.
- A *Notebook* provides a central place for storing inputs, outputs, and so forth; it is shown on page 11

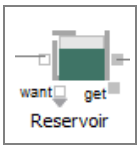
Run the example model

- Select Run > Run Simulation or click the Run Simulation button in the tool bar.

As the simulation runs, progress information will be displayed in the status bar at the bottom left of the application window. (With a small model like this, and with animation turned off, the messages may go by too quickly to be read.) Since it has been set to do it, at the end of the simulation run the Line Chart block opens and displays the graph.

Run with animation

This model doesn't have much animation in it, but the Holding Tank (labeled Reservoir) automatically animates its contents on its icon.



- Select Show 2D Animation in the Run menu or select the Show 2D Animation tool
- Run the simulation again

The level on the Holding Tank rises as the simulation runs. The minimum and maximum values for the level are set in the Holding Tank block's Animation tab.

See results

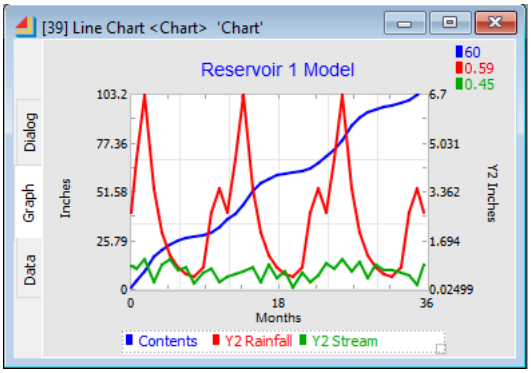
There are many ways to see information during and at the end of the simulation run. For this model, information is reported on graphs and data tables, in a notebook, in block dialogs, and through animation.

✦ See more in "Report model results and export data" on page 43.

On graphs and in data tables

ExtendSim comes with a number of flexible graph types to use in models; the Reservoir model uses a Line Chart block.

The Reservoir model runs for a simulated 36 months. While the model runs, ExtendSim displays the history of the values over time on the Line Chart's Graph tab, which by default remains open when the simulation is finished.



The Line Chart in this model has information entering three of its input connectors, so the graph displays three lines. The graph also has two value axes, each using a different scale. The graph's legend indicates which axis is used for each line (Y2 indicates the values are plotted against the right axis) and what each line represents.

In this case, the blue line, labeled 1, is matched with the left axis to show the total contents of the reservoir over time. Using the right axis, the red line (2) shows the amount of water entering the reservoir from the rainfall alone and the green line (3) shows the input from the stream. (Because the model uses random numbers for the stream, specific values may be different after each run.)

✦ To access windows for customizing how the graph is displayed and how the lines are drawn on it, right click on the Graph tab. To tear off one of the Line Chart's tabs so it is in a separate window, click and drag the tab by its name. Click a torn-off tab's close box to return it to the Line Chart.

	Contents	Rainfall	Stream
0	0.00000000	2.60000000	0.75542047
1	3.35542047	4.40000000	0.67972249
2	8.43514296	6.70000000	0.96293838
3	16.09808134	3.40000000	0.20542146

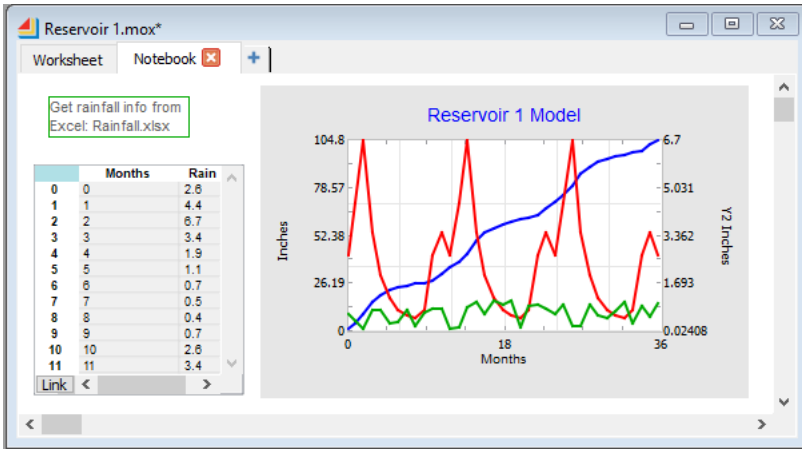
The Line Chart's Data tab shows the data points which produce the line, as seen on the left.

The Line Chart's Dialog tab has options for when to show the graph, when to autoscale, and so forth.

In Notebooks

Notebooks are customizable tear-off windows that are often used as a "dashboard" for the model - to control model parameters, report simulation results, and document the model. The

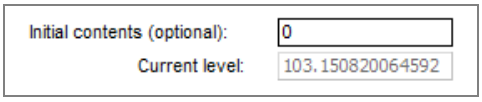
important parameters and tables from the dialogs of blocks in the Reservoir model have been cloned (duplicated) in its notebook.



✦ Cloning dialog items is discussed in the How To section of the User Reference.

From the model's blocks

Information can often be found on block icons, in block dialogs, and at block connectors. For example, when the icon of the Holding Tank block (labeled Reservoir) is double-clicked, its dialog opens and reports the current contents.



Through animation

ExtendSim provides built-in and customizable 2D animation on block icons and on the worksheet.

Using other blocks

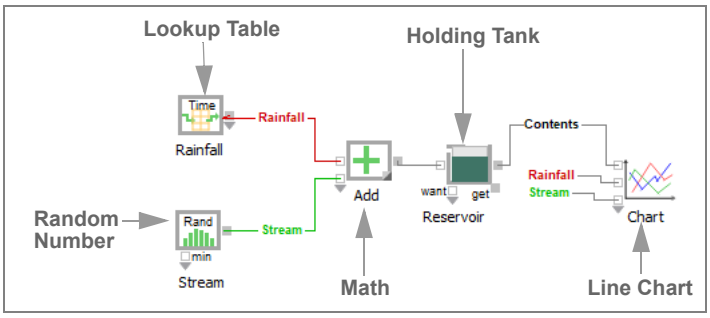
This model is so simple that it doesn't need further analysis. However, for larger and more complex models, the blocks in the Report library gather information as the model runs and display and calculate statistics on the results.

Model basics

In the simplest terms, ExtendSim models are composed of *blocks* and *connections*. This model, for example, has five blocks, as seen on the model worksheet below. As the model runs, information goes into a block, is processed and/or modified, and is then sent on to the next block, usually via a connection. Except for some hierarchical blocks (discussed on page 18), blocks are stored in repositories called *libraries*.

Blocks

Each block in ExtendSim represents a portion of the process or system that is being modeled.



Sources for blocks

The blocks for your models can come from one or more of the following:

- Pre-built libraries of blocks, for example the Value library, included with ExtendSim CP. These blocks simplify modeling tasks such as performing calculations, specifying a distribution, running optimizations or scenarios, and importing and exporting data.
- Equation-based blocks, also included with ExtendSim CP, that use formulas and/or ModL code entered in their dialogs to calculate values for, and perform operations in, models.
- Custom blocks you create the same way the blocks in the ExtendSim libraries were created—using the included ExtendSim IDE and its programming tools—then save in ExtendSim libraries for use in all models.

For more information, see “Blocks for building continuous models” on page 17 and “The Big Picture” on page 43.

Names and labels

- Blocks have *names*, such as Math or Holding Tank, that signify the function they perform. A Holding Tank block, for example, will have the same functional behavior in every model you build.
- You can also add a *label* to a block to indicate what it represents in your specific model, such as a Holding Tank block labeled Reservoir.
- Blocks come from libraries, which are discussed in “Libraries” on page 16.

✦ Right-click a block to see its Properties, such as its position on the worksheet.

Icons

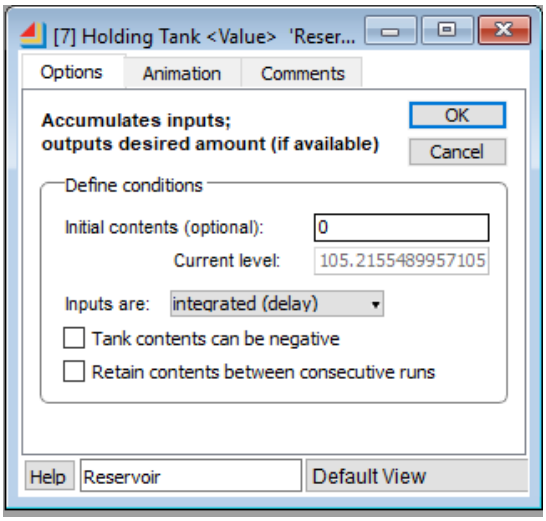
A block’s icon is usually a pictorial representation of its function. For instance in this model the block labeled Reservoir is a Holding Tank block. Its icon symbolizes a tank that can have quantities added or removed from it.

The small squares attached to the icons are *connectors*, which are discussed in more detail in “Connectors and connections” on page 15.

✦ Place your cursor over a block’s icon on a model worksheet to see a tooltip with the block’s object ID, name, and library. If *Include additional block information* is checked in the Edit > Options > Model tab, the tooltip will also include a short description of the block’s function.

Dialogs

Most blocks have a dialog associated with them. Dialogs are used to enter values and settings before running simulations and to see results during and after the run.



To access a block's dialog, double-click the block's icon or right-click the icon and select Open Dialog. For example, if you double-click the Holding Tank icon, the dialog at right opens.

At the top is the block's global object ID, its name, the name of the library it resides in (inside the angle brackets), and its label (if present). Global object IDs are unique identifiers assigned sequentially to blocks and other objects as they are placed in a model.

✦ When there are multiple models and several open dialogs, it can be difficult to remember which block comes from which model. To see which model a block is in, hover over the block's name in the dialog's title bar.

Dialog tabs

Block dialogs may have tabs on top and/or on the left side.

- Most dialogs have tabs across the top for entering values and settings, getting results, controlling animation, and so forth. As shown above for the Holding Tank block, the tabs are Options, Animation, and Comments.
- If present, tabs on the left side are for switching between windows. For example, the Line Chart's window has three tabs on the left (Dialog, Graph, and Data).

The tabs can be detached from the block to facilitate viewing the settings.

Interactive dialog items

Some dialogs calculate and display values that are generated as the model runs. If you leave a dialog open during the simulation, it will slow the run but you can watch the impact on different variables. This *interactive simulation* capability means you can even change some of the settings in a dialog during a simulation run, such as choosing different buttons or entering new values.

✦ When you click a button while the simulation is running, the block gets that changed value on the next step. However, if you type text or enter numbers into a parameter field or dialog table, the model pauses while you are typing in order to get your entire input.

Help, label, and icon view

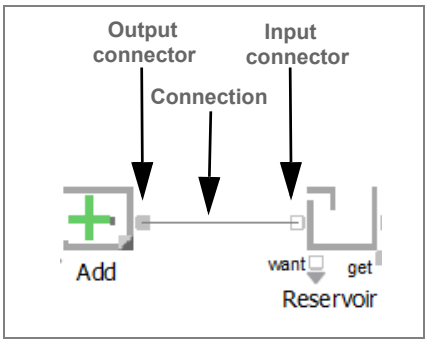


As is true for most blocks, at the bottom of the Holding Tank's dialog is a Help button. It provides information about the block, such as its purpose and use, connector usage, descriptions of each dialog item, and so on. To the right of the Help button is a text box for entering a label for that specific instance of the block in the model. The View popup is for

changing the icon's orientation or appearance when you build models—for example, the Holding Tank offers a choice of *Default View* and *Default View Reverse*.

Connectors and connections

- *Connectors* are used to get data into and out of a block during a simulation run
- *Connections* are the mechanisms for transferring data and information between blocks



Connectors

Most blocks in ExtendSim have input and output connectors (the small squares or circles attached to the block's icon). As you might expect, information flows into a block at input connectors and out of the block at output connectors.

In ExtendSim, the behavior of most connectors is predefined for each specific block. For example, if you set a Math block to use a function (add, subtract, divide, etc.) it knows what to do with the values that are input into the block. This makes model building easy since you can connect blocks and run simulations without having to write equations or logic to define what each block should do with the inputs or outputs.

A block can have many input and/or output connectors; some blocks have none. In addition, a block can have single connectors or arrays of connectors depending on how the block is constructed.

Connectors in the Value library blocks

The blocks in the Value library have connectors that get and report informational values. For example, the Math block shown here has input connectors on the left side and an output connector on the right.



- The gray outlined connectors on the left are formed by a variable connector that has been expanded to accommodate two inputs. Variable connectors act like a an array of single connectors that can be expanded or contracted to enable a required number of connections. As shown here, a variable connector is usually designated by a dark gray arrow. For instance, when the Math block is set to add its inputs, the variable connector allows for values from multiple sources to be added together.
- The solid gray output connector on the right side of the Math block is a single connector for providing outputs or results.

Expanding a variable connector

To expand the connector array for a variable connector:

- Hover over the dark gray arrow that is below the input connector on the left side of the Math block. (Don't hover over or click on the input connector, just hover over the gray arrow.) If you are in the correct spot, the cursor will turn into a dragging cursor—a double-headed up/down arrow—indicating that the array of connectors can be resized.



At left: gray arrow below connector.
At right: double-headed arrow indicating number of connectors can be changed.

- Once the cursor turns into the double-headed arrow, click and drag down until the desired number of connectors are exposed, as shown here. As you drag, more connectors appear and the number of connectors is displayed.



✦ See the User Reference for more information about variable connectors, including how to collapse them.

Connector function

To get information about a connector's function, click the Help button in the bottom left-hand corner of the block's dialog or see the User Reference.

✦ Place your cursor over a connector to see its name and current value; that value will change as the model runs. You may also see additional information depending on how the block is programmed.

Connections

The connections between blocks represent the flow of information. The simulation itself is a series of calculations and actions. In continuous process models, each calculation is called a *step*.

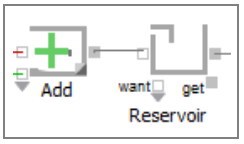
✦ In the Reservoir model, the blocks calculate in an order determined by the connections, starting at the left and going to the right.

Making connections

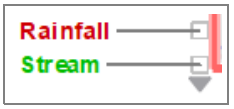
There are three ways to make physical or logical connections between blocks:

1) *Line connections* physically join the output of one block to the input of another. using connection lines

   - The Point and Click method is often used to physically draw a connection line between one block's output connector and another block's input connector (or vice versa).
   - When an ExtendSim block's connector is right-clicked, a special *Smart Block technique* can be used to add a block to the model and at the same time automatically connect it to the original block with a connection line.



2) *Named connections* use text labels as outputs and inputs, causing data to jump from the output to the input without using connection lines. In the Reservoir 1 model shown on page 13, the values for Rainfall and Stream are transmitted using named connections.



3) The *Throw/Catch* feature allows remote connections, even between hierarchical levels; this is discussed in the User Reference.

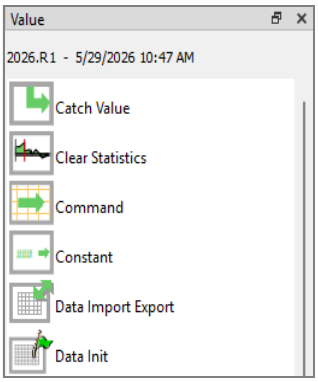
Libraries

Libraries are repositories for the blocks used to build models. When you open an existing model, ExtendSim automatically opens the libraries that contain the blocks used in the model. Libraries are also automatically opened if listed in the table in the Edit > Options > Libraries tab.

Libraries can be manually opened, closed, and created in the Library menu. Open libraries are listed in alphabetical order at the bottom of the Library menu.

✦Any data entered in the block's dialog is stored within the model, not within the library.

Library windows



Each library has an associated *library window*, listing its blocks and their names. By default when you open ExtendSim, a library window for each open library opens and docks on the right side of the application window; to change that, see the Edit > Options > Libraries tab. The name of each library window is shown on the top of the docked window and in a tab along the right side.

Library windows behave like other docking windows. For example, to detach a library window from its docked position such that it becomes a floating window, either double-click its title bar or click on its title bar and drag to where you want it. To re-dock a library window, double-click its title bar or click on its title bar and drag it back to the docking area.

Use the close box to close a docked library window. To easily close all the library windows, use the Library > Close All Library Windows command or the corresponding button in the toolbar at the top of the windows. The Open All Library Windows command and button re-opens the library window for every open library.

✦ Closing a library window does not close the library; use the Library menu to close a library.

Blocks for building continuous models

As discussed more below, to create continuous models you can:

1) Use blocks from the libraries shipped with ExtendSim
2) Create custom blocks and store them in libraries for reuse
3) Use libraries of continuous blocks developed by third parties

In addition, you can encapsulate processes or portions of the model into hierarchical blocks.

Use blocks from ExtendSim libraries

ExtendSim CP includes several libraries of pre-built blocks

Value library

You can easily build continuous models in many fields using only the Value library that is included in every ExtendSim product. These blocks are pre-programmed for data management and model-specific tasks and allow you to perform complex modeling steps often with just the click of a button. For example, select a function in a Math block that automatically takes the inputs and calculates an output, or use the Random Number block's popup menu to specify a distribution; both blocks are in the Value library.

For advanced modeling and integration with external tools, ExtendSim also provides the Python Bridge block. This block enables communication between ExtendSim models and Python, allowing you to incorporate Python-based calculations, data processing, and external libraries directly into your simulation workflow.

For added power and flexibility, use the *Equation block* to create compiled equations, from a simple operation to a full programming segment. It uses formulas and/or ModL code entered in

its dialog to calculate values for, and perform operations in, models. The User Reference has complete information about using equation-based blocks.

While the Equation block is ideal for calculations within ExtendSim, the Python Bridge block extends this capability by enabling integration with external Python code and analytics.

✦ See the Appendix in the ExtendSim User Reference for a listing and brief description of the blocks in the Value library.

Chart library

The blocks in the Chart library are useful for any type of model. The most commonly used block is the Line Chart, which traces the history of values over time during the simulation.

Report library

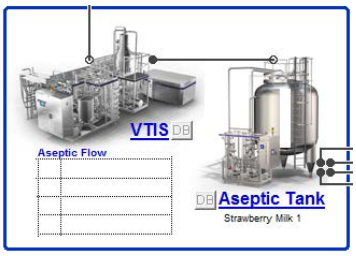
The blocks in the Report library are also useful for any type of model. They gather information as the model runs and display and calculate statistics on the results.

Utilities library

The Utilities library has several blocks that are useful for creating a custom user interface for models and for performing other tasks and activities. See “Control panel/dashboard interface” on page 45 for some examples.

Hierarchical blocks

If your model becomes too cluttered with blocks or you’d like it better organized, encapsulate portions of the model into a hierarchical block. To see the submodel, double-click the hierarchical block. Hierarchical blocks are created using simple menu commands; they can be copied from model to model and stored in libraries for reuse in other models.



Hierarchical blocks with custom icons

Create custom continuous process blocks

Continuous simulation is used in a wide range of diverse fields, and it would be impossible to supply one solution that would meet everyone's needs. To make it easy to build blocks with custom dialogs and behavior, ExtendSim contains an integrated, compiled development environment. You can create your own blocks—even your own libraries of blocks—and use them to build models. And because you have the source code for blocks that are packaged with ExtendSim, you can copy an existing block and adapt it to your needs or create an entirely new block from scratch.

Because the development environment is optimized for simulation and user-interface design, you can build blocks with less effort and more flexibility than by using a traditional programming language. The ExtendSim development environment has the functionality you need to create blocks that can:

- Process data, perform calculations, and show results in numerical, graphical, and animated form
- Communicate with other ExtendSim blocks and with external applications
- Interact with the user

By programming your own blocks you can:

- Obtain specific behaviors not available in the blocks included with ExtendSim
- Combine the functionality of several ExtendSim blocks into one custom block for increased calculation speed and convenience. (Note that this is different from using hierarchy to encapsulate several distinct blocks as a submodel within one block.)
- Develop a library of blocks for a specific discipline, such as for control systems or paper-making processes
- Design your own modeling architecture

Blocks you create can be saved in libraries and used throughout your models just as you would use any of the standard ExtendSim libraries. And the blocks that are packaged with ExtendSim, such as the Value library, are designed to work well as supplements to any custom blocks you may develop.

✦ The Technical Reference has all the information you need to program your own blocks.

Obtain third-party libraries

Third-party developers use the ExtendSim environment to create libraries of blocks customized for specific fields. For more information, go to www.ExtendSim.com.

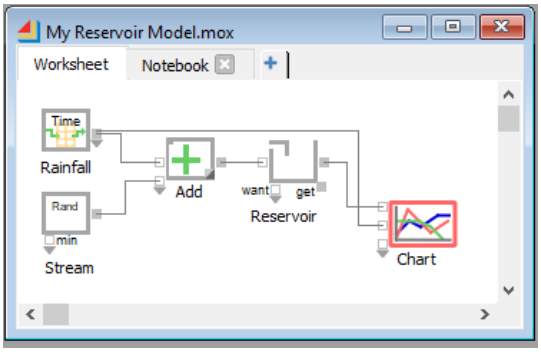
✦ Because the ExtendSim CP product does not have any discrete event capability, discrete event and discrete rate blocks built by third parties will not run with the ExtendSim CP product.

Continuous Process Quick Start

Tutorial

Overview

This tutorial shows how to build the simple continuous process reservoir model shown below.



⚠ It is important that you have read the previous chapter, or understand ExtendSim basics, before proceeding.

How to use ExtendSim CP

As you will see in this tutorial, the most common way to create a continuous process model is to connect blocks together, where each block represents some portion of the process. Then enter values in block dialogs, set the simulation parameters, run the simulation, and analyze the results.

A simple continuous process system

When building any simulation model, it is often best to start with a simplified subset of the process. Then continue adding detail, verifying and validating at each stage, until you arrive at a completed representation of the system. This will make the model building process more manageable.

About the model

This example simulates what happens to a reservoir's water level as water enters it over a 36 month period. When you have finished this chapter, your model should look like the one shown above.

✦ This model will be similar to the model you opened and ran in the previous chapter.

Assumptions

- There is no water in the reservoir at the start of the simulation
- No water is removed from the reservoir
- As water sources add their contributions each month, the water level rises
- There is a table of data for the expected amount of rainfall each month

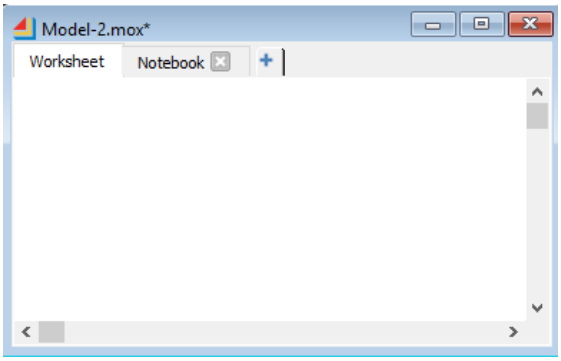
- The stream adds between 0 and 1 inches per month
- The model runs for a simulated time of 36 months

Launch ExtendSim

By default, when you launch ExtendSim the Getting Started model opens as well as the major ExtendSim libraries and their library windows.

Open a new blank model

▸ Use the toolbar button or the File menu to open a new model.



By default a new model window, named Model-X, opens.

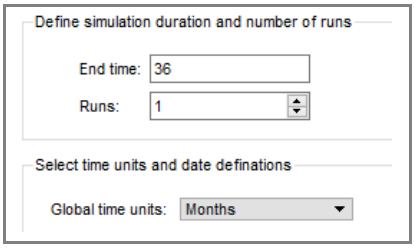
The Chart and Value libraries and library windows should automatically open due to settings in the Edit > Options > Libraries tab. If those libraries aren't open, manually open them from the Library menu.

⚠ Some ExtendSim packages contain an Item library. If ExtendSim also opens the Item library, new models will open by default with an Executive block on the worksheet. **Continuous models cannot contain an Executive block.** If your new model contains an Executive block, delete the block. To prevent an Executive from being placed in new continuous process models, change the default option in the Edit > Options > Model tab.

Set the simulation parameters

▸ Select the command Run > Simulation Setup (Ctrl/control J)

This command opens a window for entering global settings for the model, such as how long and how many times the simulation will run.



The most common simulation settings you will need to enter in the Simulation Setup window (and often the only ones) are located on the Setup tab: the *End time*, the number of *Runs*, and *Global time units*. For most purposes, you want the simulation to start at the beginning, so you would use the default start time of 0.

Use the model assumptions to set up the simulation run:

▸ In the Setup tab, enter the simulation parameters:

  ▸ **End time: 36**

  ▸ **Start time: 0** (default)

  ▸ **Runs: 1** (default)

  ▸ **Global time units: Months**

▸ Leave the other Simulation Setup settings at their defaults

▸ Click **OK** to close the window and save changes

With these settings, the model will run for a simulated time of 36 months.

Save the model

▸ Choose File > Save Model As and name the file **My Reservoir Model**.

✦ Models with changes that haven't been saved have an asterisk (*) at the end of their name in the title bar. Each time you save a model, ExtendSim creates a backup, *ModelName.bak*, of your previously saved model. To recover using a backup file, rename the file to *ModelName.mox*.

Blocks used for the simple Reservoir model

The following table lists the five blocks used in the model.

Name (Label)	Library	Block Function	Purpose in Reservoir Model
Lookup Table 	Value	Acts as a lookup table. You can choose to set it to output data based on the current time or based on the value it receives at its input.	Represents rainfall entering the reservoir each month. The amount of rainfall is based on historical averages and varies with the month.
Random Number 	Value	Generates random integers or real numbers based on the selected distribution.	Generates random values that correspond to the changing flow of the stream. In this model, the stream increases the reservoir level between 0 and 1 inch of water per month.
Math 	Value	Performs mathematical functions. The same Math block can be used for a wide variety of purposes by selecting the desired function from a popup box in its dialog.	Adds the amounts from the two different water sources and transfers them to the Reservoir.
Holding Tank 	Value	Accumulates the total of the input values. It also allows you to request an amount to be removed and outputs that requested amount, if available.	Accumulates water from its two sources. In this model, the tank has an infinite capacity and nothing is removed.
Line Chart 	Chart	Displays graphs and tables of data input from continuous blocks.	Shows the amount of input from the two sources and the level of water in the reservoir as it is affected by the amount of water entering it.

There is nothing fundamentally different about the structure of these different blocks. Any block may create, modify, or present information, and many blocks perform more than one of these functions. You can, of course, have multiple instances of the same block within a model.

Placing and connecting blocks on the model worksheet

Methods for placing blocks in a model

There are two main methods for placing and connecting blocks in a model:

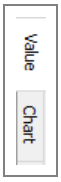
1) The Point and Click method is useful for adding any block anywhere on the model worksheet:
   - Click a block in the Library menu or library window to select it
   - Click somewhere on the model worksheet to place it
   - As needed, connect it to another block using one of the connection methods discussed on page 16
2) The Smart Block method is a quick way to add a new block to a model and connect it to an existing block at the same time:
   - On the model worksheet, right-click on one block's connector and, from the list, select the new block to add. This will place the new block next to the original and connect them.

You will use both methods for this tutorial. However, to use the right-click technique, a block must already be on the worksheet to start the connection chain. So you will place the first block on the worksheet using Point and Click.

Use Point and Click to place a Value library block on the worksheet

Lookup Table block

To use Point and Click to place a **Lookup Table** block (Value library) on the model worksheet:



- Bring the **Value** library window forward by clicking once on its name in the list of library windows on the far right of the application window
- In the Value library window, click once on the **Lookup Table** block's icon
- Then click on the model worksheet wherever you want the block to be placed



✦ If you select the wrong block, click the Esc (escape) key to unselect it. (You can also place multiple instances of the same block on a worksheet by holding down the Alt or Option key while repeatedly clicking on the worksheet.)

Use the Smart Blocks technique to place the remaining blocks

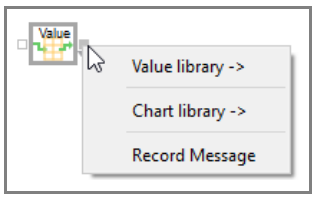
Now, because it's an easy and fun way to add blocks to a model, use the Smart Block technique to place additional blocks on the worksheet.

⚠ Since Smart Block involves right-clicking on a connector, you must already have at least one block on the model worksheet for it to work. This technique works with the major ExtendSim libraries and, if they have been programmed to use it, with custom libraries.

Math block

- Go to the model worksheet.

- Right-click on the Lookup Table's **output connector** (the solid gray square on the right of the icon). A popup with the two libraries (Value and Chart) you would most commonly use appears.
- Select the **Value library** in the popup. This opens a new popup listing all the blocks in the Value library.
- Select the **Math** block from the popup list of Value library blocks.



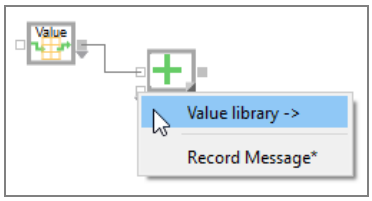
This automatically places the Math block on the worksheet and connects its top input connector to the output of the Lookup Table block.

✦ While not needed here, the Smart Block technique can also be used to connect a second block to an output (hold down the Ctrl/Cmd key while right-clicking on the output) or to insert a new block between two connected blocks (separate the two blocks to provide room for the new block, then right-click on the output or input of one of the existing blocks to insert the new block).

Random Number block

Blocks can also be added by right-clicking on an *input* connector.

- Right-click on the *Math* block's second **input** connector, the input connector that is not yet connected
- Select the **Value** library
- In the Value library window that appears, click once on the **Random Number** block



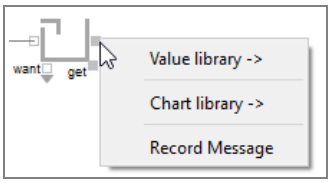
This adds the Random Number block to the model, connected to the second input of the Math block.

Holding Tank block

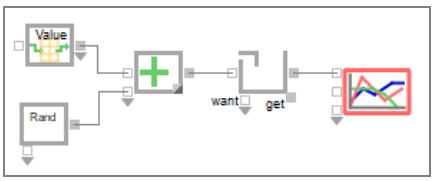
- Next, right-click on the *Math* block's **output** connector
- Again select the **Value library**
- Choose the **Holding Tank** block

Line Chart block

- Finally, right-click on the unlabeled **output connector** that is on the right side of the Holding Tank's icon
- Select the Chart library in the popup. This opens a new popup listing all the blocks in the Chart library.
- Select the **Line Chart** block.



At this point, your model should look like the one shown here.



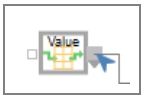
Save the model

- Save the model to save your changes.

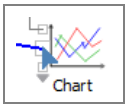
Make connections and adjust the blocks

As discussed in “Making connections” on page 16, there are multiple ways to make connections between blocks. For simplicity, this tutorial uses line connections.

Connect the Lookup Table block to the Line Chart



- Hover the cursor over the **Lookup Table**’s *output* connector until that connector enlarges.
- Click once on that output connector.
- Then move the cursor and hover it over the **Line Chart’s** second *input* connector. That input connector enlarges.



- When the connection line turns thick and blue, click once on the Line Chart’s input connector to make the connection.

This connection line will cause the information from the Lookup Table block to be reported at the Line Chart block’s second input connector.

✦ Connection lines will turn solid dark gray when properly connected; they appear as a dashed red line if the connection has not been made correctly.

Adjust connections (optional)

By default, connection lines are drawn using a Smart Connection technique that provides intelligent connection line control when moving blocks around. If Smart Connections don’t result in a good appearance for the finished model, you can change the connection line style and adjust the line.

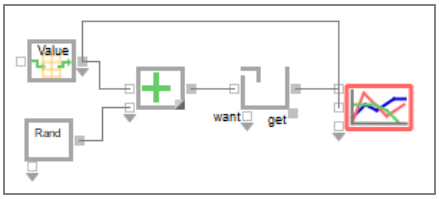
- To change the appearance of an existing connection line:
  - Right-click on the connection line
  - Choose the Right Angle or Free Form Connection
  - Move the line segments as desired by selecting and moving the anchor points, the small squares at each joint.
- Reposition the blocks as desired.

✦ To change connection line behavior and appearance in a model, go to the Model > Connection Line Style menu or use the right-click menu. To change the default connection line behavior for all models, use the Edit > Options menu.

Save the model

- Save your changes.

When finished, the model should look similar to the one shown here.

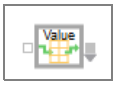


Enter dialog parameters and settings

To reflect the basic assumptions for this model, the parameters need to be entered.

Lookup Table block that represents rainfall

The assumptions state that there is a table of data for the expected rainfall for each month. In a real-life situation, for example, these numbers may have been determined by annual recorded averages.



- To open the dialog of the Lookup Table block, double-click its icon or right-click on the icon and choose Open Dialog.

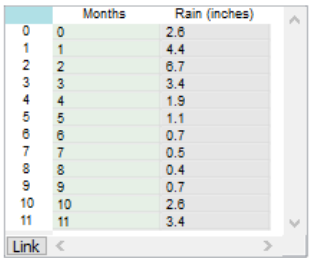
By default, the block is set to *Lookup the: input value*. In that mode, the block outputs a value that corresponds to the value it receives at its input. However, you want the Lookup Table block to output a value—the amount of rainfall for the month—that corresponds to the current simulation time.

Enter settings in the Table tab

- Customize the dialog's *Table* tab by setting the behavior of the block to look up simulation time each month:
  - **Lookup the: time.** This setting means the block will compare current simulation time to a time in the table and output the corresponding value.
  - **Output is: stepped**. This means that ExtendSim will use the exact values you enter in the table, not an interpolated amount.
  - **Time units: months** (the model default as set in the Simulation Setup command)

Enter data in the data table

- Increase the number of rows so the table has room for twelve months of data:
  - Click the +/- resize button - the green square at the bottom right of the table
  - Enter 12 for the number of rows and 2 (the default) for the number of columns
  - Click OK



	Months	Rain (inches)
0	0	2.6
1	1	4.4
2	2	6.7
3	3	3.4
4	4	1.9
5	5	1.1
6	6	0.7
7	7	0.5
8	8	0.4
9	9	0.7
10	10	2.6
11	11	3.4

Link

- Enter data into the table as shown in the screenshot at right. *Or copy the data from the Rainfall.txt or Rainfall.xlsx files located at ExtendSim/Examples/Tutorials/Continuous and paste it into the data table.* Note that since the starting time is 0, the first month's index is 0.

✦ After entering a value, tab or use the keyboard's arrow keys to move to the next cell.

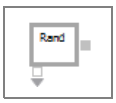
- At the bottom of the Table tab, check the checkbox labeled **Repeat table every** and enter **12** in the months box. This causes the determination of monthly rainfall to restart every 12 months.

With this information, the table's left column will specify the month and the right column will specify the amount of expected rainfall in inches. At each step, ExtendSim will check the block's table for a time in the left column that is less than or equal to the current simulation time and output the corresponding value (inches of rainfall) from the right column.

Options tab

- On the block's *Options* tab, enter the following text in the "Column labels" box to give more meaningful headings to the table (be sure to include the semi-colon): **Month;Rainfall (inches)**
- Enter **Rainfall** as the label for the block
- Click OK to save changes and close the block's dialog.

Random Number block that represents water from the stream



In this model, the Random Number block is used to specify a random distribution of water entering the reservoir from the stream. The distribution is a real number between 0 and 1, indicating that the stream will add between 0 and 1 inches of water to the reservoir's level each month.

- Open the Random Number block's dialog.
- It happens that the settings are the same as the assumptions for this model:
  - **Distribution: Uniform Real** (default)
  - **Minimum: 0** (default)
  - **Maximum: 1** (default)
- Label the block **Stream**.
- Click OK to close the dialog.

Math block



As indicated by the plus sign on its icon when you placed the block in the model, the Math block is set by default to add its inputs.

- Label the block **Add** and close its dialog.

Holding Tank block that represents the reservoir



The Holding Tank block represents the level of water in the reservoir. The assumptions state that the Holding Tank has no beginning contents and does not release any of its contents.

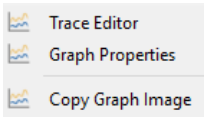
- Open the Holding Tank block's dialog.
- In the Options tab, set:
  - **Initial contents: 0** (default)
  - **Inputs are: integrated (delay)**
- Label the block **Reservoir**.
- Click OK.

✦ Since the values are based on time, the Holding Tank should be set to integrate, rather than sum, its inputs. This will output the value at time 1 that has been calculated for the period from time 0 to 1. Integration for continuous simulations is discussed more thoroughly on page 38.

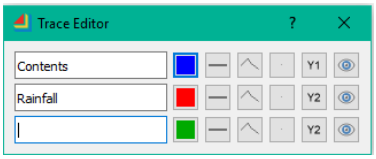
Configure the Line Chart block

As this model is constructed, the Line Chart will report the level of water in the reservoir.

Trace Editor



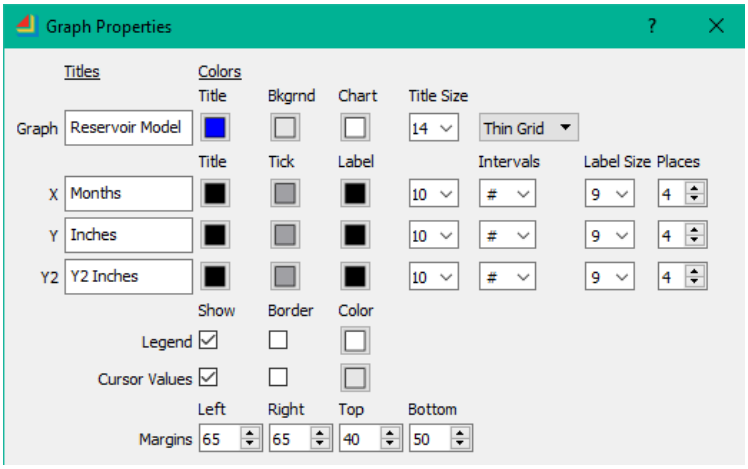
- Open the Line Chart block by double-clicking its icon or by right-clicking and choosing Open Dialog.
- Right-click on the block's Graph window. The two options at the top of the popup are Trace Editor and Graph Properties.
- Choose the Trace Editor. This is where you specify how the lines are drawn on the graph.



  - Name the first trace **Contents**
  - Name the second trace **Rainfall** and choose that it is plotted on the **Y2** axis
  - Close the Trace Editor
  - Notice that the new labels appear in the key on the graph. You can resize the key and place it anywhere you want on the graph.

Graph Properties

- Right-click the Graph window again and this time choose Graph Properties. This is where you customize the titles, colors, and other properties of the axes, as well as characteristics of the numbers that are displayed.



  - Name the graph **Reservoir Model**
  - For the title of the X axis, enter **Months**
  - For the title of the Y axis, enter **Inches**
  - For the title of the Y2 axis, enter **Y2 Inches**
  - Close the Graph Properties window

✦ You can also enter titles directly on the graph—click on the text and change it, then press the Enter key.

Dialog tab

- Click the Dialog tab on the left side of the graph. The tabs in the Dialog are for labeling the block and for setting what happens to the graph during or after the simulation run. In the Dialog:

- In the block's Display tab, choose if you want the graph to not open, or to open at the beginning or the end of the simulation run. Also choose when you want the graph to autoscale.
- At the bottom of the dialog, label the block **Chart**.
- Close the block's dialog.

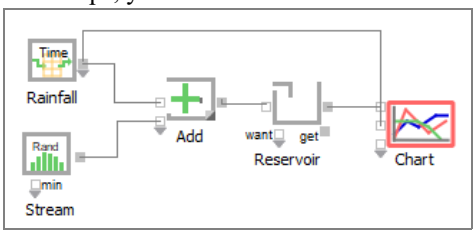
✦ See the Line Chart's Help button or the User Reference for more information about the tools and dialogs that control this block.

Save the model

- Save the model to save your changes

Your tutorial model

If you have followed all the steps, your model should look similar to the model shown below.



Running a simulation

Before running the model, decide if you are going to run a single model one or more times, run different models all at the same time, or run copies of a single model in parallel by deploying multiple instances of ExtendSim.

Run modes

There are two modes toggled using the Run Mode toolbar button:

1) To run a single model as quickly as possible, even if you are running that model multiple times, use the fastest run mode as shown here. This is the default and what you will use for this tutorial.
2) To run different models all at the same time, use the multi-threaded run mode . This takes advantage of the ExtendSim multi-threading capability—the user interface is processed in one thread and the processing for each model occurs in separate threads. In this mode, the Run Simulation button changes to .

Multi-core Runs

To run a single model multiple times by spawning multiple instances of ExtendSim on a computer's multiple cores, use the Multi-core Analysis Run button on the toolbar.



See "Multicore Analysis" under Running a simulation in the How To chapter in the ExtendSim User Reference for details on how to set up and run models on multiple cores.

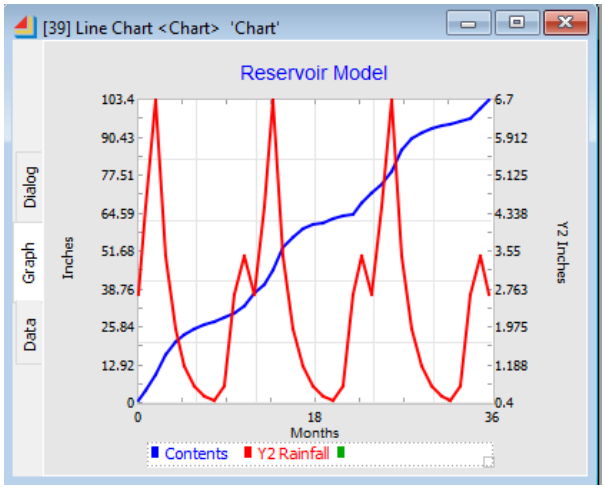
Run the simulation

- Click the Run Simulation button in the tool bar or choose the Run > Run Simulation command or right-click to run the simulation.

Verify and validate results

This is a good opportunity to verify and validate results.

- If it isn't open, double-click the icon of the Chart block. The block shows the results as a graph on its Graph tab, seen at right, and as data in its Data tab.



As configured, the graph monitors the total amount of water in the reservoir on the Y axis and the inches of rainfall on the Y2 axis.

Given how the model is constructed and the settings in its dialogs, are the numbers in the graph what you expected to see? Does the model accurately represent the system being modeled?

✦ See the User Reference's Process of Simulation chapter for information about verifying and validating a model.

Save the model

- Save the model; you will use it in the next chapter.
- To compare your model to ours, see the model named *Reservoir 1* in the folder Documents/ExtendSim/Examples/Tutorials/Continuous.

Note that the Reservoir 1 model that we built uses named connections to plot the values coming from the two water sources—Rainfall and Stream. Your model should get similar results.

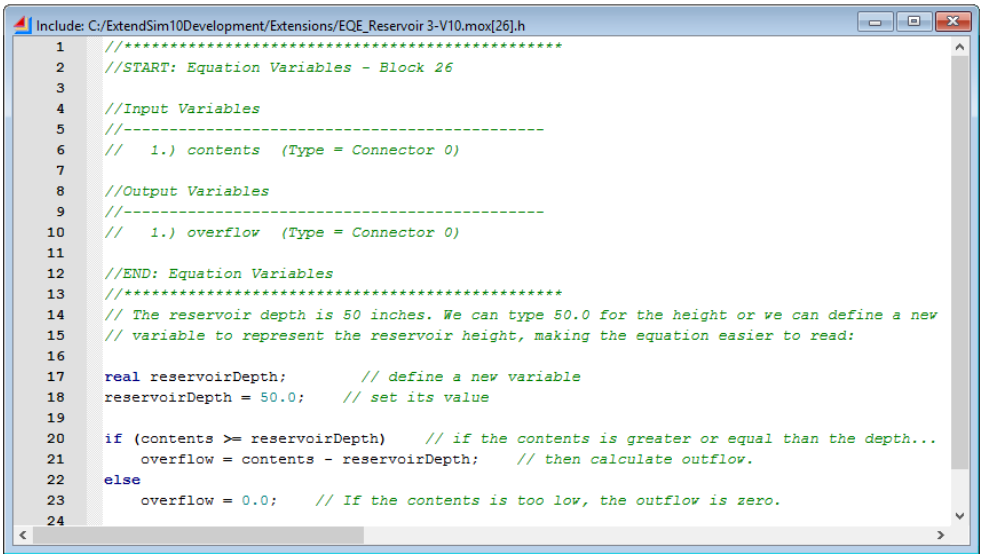
Additional models

Reservoir models

There are two additional reservoir models located at Documents/ExtendSim/Examples/Tutorials/Continuous. These models show how to model a limitation on the reservoir's capacity and the resulting overflow.

1) The Reservoir 2 model does the calculations using logic blocks from the Value library. It also has a button (About This Model) that explains the model. The "button" is actually a hierarchical block that uses the Buttons block (Utilities library) to open the hierarchical block when clicked.

2) Reservoir 3 does the exact same calculation as Reservoir 2 using one Equation block (also from the Value library). That Equation block's equation editor is shown below.



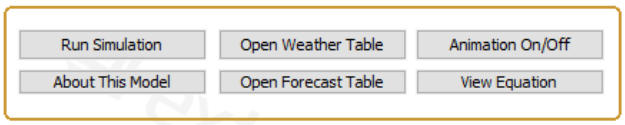
Other continuous process models

Additional continuous models process models are located at Documents/ExtendSim/Examples/Continuous.

Standard Block Models folder

These models have been built with the blocks that ship with ExtendSim CP:

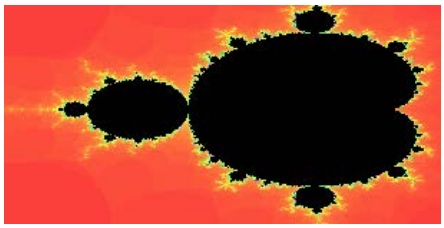
- Differential Equation
- Drug Ingestion
- Markov Chain Weather
- Monte Carlo
- Noisy FM System
- Predator Prey



Custom Block Models folder

To display some of the ExtendSim programming capabilities, these models have been built using custom blocks created in ExtendSim CP:

- Fish Pond
- Mandelbrot
- Planet Dance



What's next

- The next chapter in this Quick Start Guide discusses some concepts specific to continuous process modeling.

- The final chapter lists ExtendSim features that will help you get the most benefit from your models.

Continuous Process Quick Start

Concepts & Terminology

This chapter discusses some concepts specific to continuous process modeling and provides tips to keep in mind when building continuous models.

✦ Concepts applicable to all types of models, such as deciding whether a model should have random elements, are discussed in the User Reference.

Simulation timing

Most simulations run for a specified time. ExtendSim determines the *duration* of a simulation run based on the values entered in the Run > Simulation Setup > Setup tab; the duration is the period from the start time to the end time.

In continuous simulations, the duration is divided into intervals or *steps* of equal length, where start time is the first step and end time is the last step. The length of time, in time units, for each step is known as *delta time* or *dt*. The delta time setting determines how frequently model data is recalculated.

As the simulation runs, simulation time advances from start time to end time at delta time per step, calculating model data at each step. At the first step, ExtendSim calculates what the status of the model is initially. Then it calculates the changes that take place over the next time step and determines a new set of data points. Model data is generated as a string of successive points corresponding to the steps in time.

✦ For each step, data is calculated for the entire period from that step up to, but not including, the next step.

Continuous simulations require that either the number of steps or the time per step be specified. As discussed below, in most cases a delta time of 1 is adequate. However, for model accuracy it may be necessary to set a different delta time.

Delta time and step size

Delta time (dt or Δt) literally means the change in time. It is defined as the length of the time interval between the present time and one time interval later. When you select a delta time, you are selecting how finely the total simulation time will be sliced up, i.e. how short the intervals between calculations will be and thus how frequently the computations will take place. Delta time is set in the Run > Simulation Setup > Continuous tab.

For the simulation results to be correct, the delta time for a continuous model needs to be small enough to accurately reflect changes that occur in different parts of the model. While a delta time of 1 is often adequate, for simulation speed or model accuracy it may be necessary to set a delta time other than 1.

Delta times other than 1

Although 1 is the default setting in the Continuous tab, delta time can be set to any number. For example, to cause the model to run faster and perform calculations less frequently, delta time could be set to 2 or any other number larger than 1.

A delta time less than 1 reduces the step size, causing calculations to be made more frequently. It also results in more steps, so that the simulation takes longer to run in real time for the same simulation time.

If delta time is not 1, it is most common that it would be less than 1. There are many reasons why delta time would need to be less than 1. Feedback loops and stiff equations in the blocks can require a smaller delta time to ensure that all calculations are reflected in the graph. Simulations that are run with too large of a delta time often show values jumping from very high to very low. This is known as instability or artificial chaos. Examples of models where a delta time of less than 1 may be required are:

- Signal processing models need to have their specific time per step (dt) either entered in the Simulation Setup dialog, or have it calculated by blocks in the model. For example, the Filter blocks calculate the stepsize based on their entered parameters.
- Differential equation models (those with integrators in feedback loops) may need to have a time per step (dt) or number of steps other than 1.
- If you are building custom blocks in process control models, you might set up the blocks so that they have a Stepsize message handler that can calculate the value for the DeltaTime variable, automating this process. See the Electronics library for some examples of blocks that do this.

✦ If a model contains the Holding Tank block (Value library) and delta time is not exactly 1, you may need to change the Holding Tank to integrate its inputs, as discussed on page 39.

Determining which dt to use

To determine what delta time setting is reasonable, first run the simulation with a delta time of 1 (the default setting). Then run the simulation with a delta time of 0.5 (one half of the original setting). If there is no significant difference between the two graphs, then delta time of 1 is appropriate. If there is a significant difference, reduce the delta time to 0.2 and run the simulation again. Continue halving delta time until you determine a delta time which results in no significant differences compared to the smaller delta time. The main idea is to use the largest delta time that will give accurate results without slowing down the simulation unnecessarily.

Specifying dt or the number of steps

ExtendSim requires that either the time per step (delta time) or the number of steps be specified in the Run > Simulation Setup > Continuous tab. You can enter delta time directly in the dialog or you can enter the number of steps and ExtendSim will calculate the delta time for you.

A value for the number of steps is automatically calculated based on the setting entered for Time per step (dt). It is computed as: floor(((EndTime-StartTime)/DeltaTime) + 1.5). You can see this by choosing the Number of steps radio button after changing the Time per step (dt).

You can also determine the granularity of the simulation run by manually entering a value for Number of steps. In most cases, this would be a number equal to the duration (length of the simulation run) so that the model calculates values once for each step, and each step would be one time unit long. A default value for delta time is automatically calculated based on the number of steps you enter; it is computed as (EndTime-StartTime)/(NumSteps - 1). You can see this by choosing the Time per step (dt) radio button after changing the Number of steps.

Example of setting the end time when delta time is 1

Assume you want a continuous simulation to run and calculate values each year for 40 years where start time is 0 and dt is 1. The value you enter for the end time depends on whether there is integration in the model.

- If there is no integration in the model, set the end time to 39. Data will be calculated at each of the 40 steps, starting at step 0 and ending at step 39. Each step's calculation would represent data for the entire year, the period beginning at that step and continuing up to but not including the next step (one delta time unit). Thus the model would calculate 40 years worth of data. If you specified start time as 1 and end time as 40, the duration would also be 40 years.
- If there is integration in the model, set end time to 40. Data will be calculated (but not output) at each of the 41 steps, starting at step 0 and ending at step 40. Each step's calculation would represent data at the beginning of the year, the period beginning at that step. However, blocks that integrate take their inputs at one step and output their results at the next step. Thus the model would calculate 40 years worth of data. If you specified start time as 1 and end time as 41, the duration would also be 40 years.

✦ Integration is discussed on page 38.

Example of setting the end time when delta time is other than 1

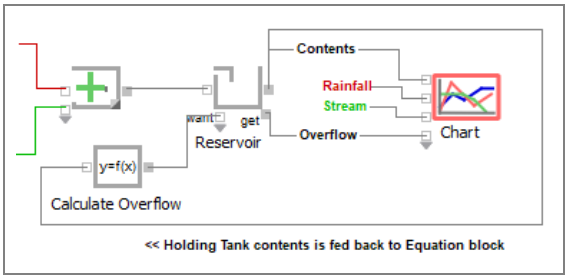
If you specify delta time as 0.5, the start time as 0, and the end time as 39, the simulation will run from time 0 to time 39 calculating data for 79 steps, each of which is one half time unit long.

If you specify a delta time which will not result in the duration being divided into equal segments, ExtendSim will adjust the end time to a higher value. This is to insure that the segments are of equal duration, and that the simulation will end at the end time displayed in the dialog. Alternately, you can select a new end time or delta time. For instance, assuming end time is 39, you could specify the Time per step (dt) as 2 years. Data could be calculated every other year, from 0 through 38; however, the last step of the simulation (step 39) would not be calculated. In that case, rather than omit a step, ExtendSim will adjust the end time to 40. This means that model data will be calculated once every two years, starting at time 0 and ending at time 40, for a total of 21 steps.

Feedback and delays

Sometimes it is necessary to create a model that tries to compensate for, or vary itself to match, changes in its inputs. For example, a public address amplifier should output the same sound level even if the speaker's voice varies in loudness. A good technique to accomplish this involves feeding back some of the amplifier's output to control its input, so that the amplifier can effectively monitor itself.

Feedback



The practice of connecting an output of a model back to an input is called feedback; it is the main factor that causes complex behavior in continuous models. A model with feedback will cause the result of a calculation to be fed back to one of the original variables in the calculation, influencing its own rate of change. Thus feedback causes iterative changes to model variables as the simulation runs. The initial output variable ripples through the calculations such that the effect loops back and re-affects the initial variable.

Delays in feedback loops

ExtendSim is designed to facilitate feedback in models, but using feedback correctly entails some additional modeling techniques. Since continuous models are recalculated at every time step, there will be a delay between when the initial variable first affects the model and when the chain of calculations ultimately and indirectly affects the initial variable. The introduction of a delay can cause the behavior of a model to change significantly, because there is a difference between the effect that instantaneous access can cause and the response that would happen if access is not instantaneous.

When you run a model that uses feedback, the feedback is delayed by one simulation time step. This happens because the output feedback signal has to be calculated before it can be used on the next simulation step, effectively delaying it for one time step. When feedback is delayed too much, it can have a deleterious effect as it arrives too late, causing the system to over- or under-compensate. Reducing the time step or delta time of the model will reduce feedback delay and result in a model with more accurate results. See “Determining which dt to use” on page 36 for more information.

Integration

Integration is a method of estimating the present value by estimating the past and/or future values of the inputs, calculations, and outputs in a model. It is another way of summing or accumulating values when the input is a rate (expressed as units per time). The advantage of integration over just summing values is that integration will accumulate correctly when delta time is not exactly 1.0, making it especially useful in feedback models that need a smaller delta time to work well. A simple form of integration (Euler) just multiplies the input rates by delta time and accumulates the result.

The Holding Tank and Integrate blocks in the Value library have integration capabilities.

- The Holding Tank block accumulates and, if wanted, releases values. As discussed below, its dialog gives the option to either sum or integrate the inputs.
- The Integrate block is used to perform the mathematical function of integration in models.

You can also use ModL functions to add integration to blocks you create.

✦ There is no feedback in the My Reservoir model you built in the tutorial and it uses a delta time of 1. For that model, the Holding Tank could be set to either sum its inputs or integrate them (but the end time would need to be adjusted). The Reservoir 3 model, located at ExtendSim/

Examples/Tutorials/Continuous, has feedback and uses a smaller delta time, so the inputs to the Holding Tank must be integrated. See below for more information.

Integration vs. summation in the Holding Tank block

he Holding Tank block (Value library) has integration capabilities. The block's dialog gives the option to either sum or integrate its input. In general, a good rule of thumb is:

- Integrate when the value going into the Holding Tank is based on simulation time units, such as a rate. For example, you would integrate when the Holding Tank block's input represents dollars per year or gallons per hour.
- Sum when the value is to be added to the block at each step or dt calculation. For example, you would sum when the Holding Tank block's input is orders or people.

Summing adds the given input to the total at each step, regardless of the time units. Integration considers the input to be spread evenly over each time unit; at each step integration adds a portion of the input to the total. For example, the following table shows the effect of inputting $2000 to the Holding Tank block when the time units are in years and dt (delta time) is set to 0.25 (for 1/4 of the year).

As seen in the table, if the Holding Tank is set to sum its inputs, it would be the equivalent of adding $2000 to the account every quarter. If the Holding Tank is set to integrate, it would be the equivalent of adding $500 per quarter.

Time	Step	Summed	Integrated (delay)	Integrated (no delay)
0	0	2000	0	500
0.25	1	4000	500	1000
0.50	2	6000	1000	1500
0.75	3	8000	1500	2000
1	4	10000	2000	2500

- *Summation* occurs at each step so there is an amount calculated at time 0. Since summation treats its input as an amount, the entire 2000 is added at each step.
- The *integrated (delay)* choice treats its input as a rate and calculates a new result at the next step. Because this backward Euler integration occurs during the interval between steps, there is no amount at time 0.
- The *integrated (no delay)* choice also treats its input as a rate. However, it calculates a new result at the current step. Because its integration occurs at each step, there is an amount at time 0.

It is also important to note that if you subsequently change the delta time to something other than 0.25, the total amount in the summed Holding Tank would be different from the amounts shown above, but the total amount in the integrated Holding Tank would remain the same.

Your choice of integration methods depends on the model:

- You would generally use the *integrated (delay)* choice when there is only one integrating block in the model, when the integrating blocks are not interdependent or cross-coupled, or when there is no feedback.
- In models with more than one integrating block, where the integrating blocks are interdependent or cross-coupled, the feedback between the blocks is usually a correction factor. If this feedback is delayed, the system may correct too late or over correct, causing the model results to become unstable. This is often observed as a graph where the traces oscillate with increasing magnitude as time progresses. The *integrated (no delay)* choice compensates for the feedback delays by outputting results one step earlier. For example, the Predator_Prey model is an example of interdependent Holding Tank blocks; the model is located in the folder Examples/Continuous/Standard Block Models.

Simulation order

The Run > Simulation Setup > Continuous tab allows you to choose the order in which ExtendSim executes block data for continuous models. The choices are Flow order (the default), Left to right, and Custom. To see the order in which blocks are executing, select the command Model > Show Simulation Order before the model is run.

✦ It would be unusual to change the simulation order from the default choice, Flow order.

Flow order

During a simulation, the blocks that compose an ExtendSim model perform calculations that generally depend on their inputs. After doing their calculations, the blocks set their output connectors to the results of that calculation so that other blocks may use their results.

In this type of system, there has to be a “first” block: a block that calculates before all of the others that depend on its results. After the first block calculates, the other blocks should calculate in the order and direction of their connections. This order is repeated for every time step of the simulation. To see this order, choose Model > Show Simulation Order.

The following are the rules that ExtendSim uses to derive the order of the block calculations in continuous models:

- Blocks that generate inputs to the simulation go first. For example, Lookup Table or Constant blocks with only their outputs connected to inputs of other blocks would be put first.
- Next, ExtendSim executes blocks that are connected to those first blocks, in the order and direction of their connections.
- Unconnected blocks and bi-directional network blocks (that have only inputs connected to inputs) are executed in left-to-right order.

✦ The Feedback block (Utilities library) is useful when there are flow-order issues due to feedback in a continuous model.

Left to right order


‘Use caution when changing from Flow order.

If you choose this option, ExtendSim looks at the left/top corner of each block on the worksheet. The left-most block gets executed first, and the next left-most block gets executed second, and so on. Blocks with equal left edges get executed in top to bottom order. If your model flows to the right, and then continues at the left below that flow, this choice will still calculate

all the left-most blocks first. If you use this order, be sure that blocks that calculate values are to the left of the blocks which need those values. Otherwise, there will be a one step delay in calculating the values.

Custom order

⚠ ‘Use caution when changing from Flow order.

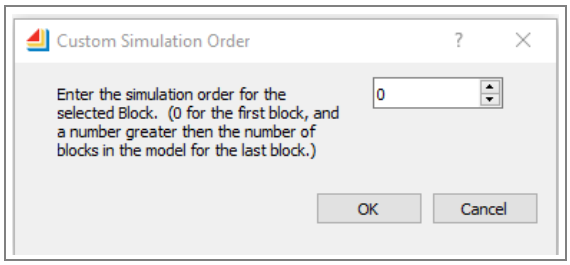
In continuous modeling, there are some situations with multiple feedback loops that do not automatically settle into the desired flow order solution. This occurs because there are multiple solutions that solve the DAG (Directed Acyclic Graph) ordering problem, and it is possible for the less desired solution to be picked.

To solve this ordering problem, start with the model set to Flow order and then change the order for a few selected blocks using the Model > Set Simulation Order command.

✦ To use the Custom order option effectively, the Model > Show Simulation Order command should be checked so that the user can see which blocks need a changed order.

⚠ The Set Simulation Order command is not enabled unless Custom order is selected in the Simulation Setup dialog and a block is selected on the model worksheet.

In the Set Simulation Order dialog, you can enter a new order number for the selected block and it will be moved into that position in the simulation order.



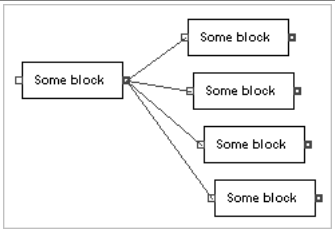
Note that any custom order is lost if either Flow order or Left to right order is selected again.

Mixing block types

You can use continuous blocks, such as those from the Value library, in discrete event and discrete rate models. You cannot, however, use discrete event or discrete rate blocks in continuous models. All blocks in the discrete event and discrete rate libraries require an Executive block that changes the timing of the model to event timing.

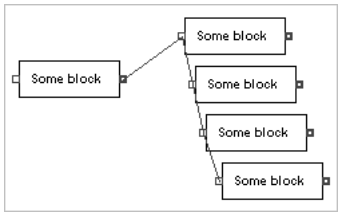
Connections to multiple inputs

In a continuous model, you can connect from one output connector to as many input connectors as you want. Multiple inputs from one output is useful when many blocks need the results of a preceding block. For instance, if you have one output connector connected to four input connectors, the model might look like the image at right.



One value output connected to four inputs in a continuous model

For aesthetic reasons, you may want to only have one connection coming from the output connector, for example when the blocks are far away from each other. Instead of four connections, connect to one of the input connectors, then connect the input connectors together (sometimes called *daisy-chaining*) in the image at bottom right. (Note that this method may be confusing to others trying to understand the model.)



Four value inputs connected together

Continuous Process Quick Start

The Big Picture

Overview

The tutorial chapter showed how to build a simple continuous process model in ExtendSim. However, if every system were as easy to model, you wouldn't need simulation! This chapter points out ExtendSim features that will help you to:

- Use equation-based blocks
- Build custom blocks
- Represent the dynamics of the system you're modeling
- Create a user interface for the model
- Get data into the model
- Manage data for use in the model
- Perform analysis and assessments
- Report model results and export data
- Communicate with other applications and devices



This chapter just lists ExtendSim major features and capabilities; the **User Reference** explains how to use these features, the **Technical Reference** shows how to create custom blocks, and example models show you how to employ them in your models.

Using equation-based blocks

```
real reservoirDepth;         // define a ne
reservoirDepth = 50.0;     // set its value

if (contents >= reservoirDepth)     // if t
    overflow = contents - reservoirDepth;
else
    overflow = 0.0;     // If the contents
```

Code in the Equation block

Based on formulas and ModL code entered in their dialogs, the equation-based blocks calculate values for, and perform operations in, models. There are six equation-based blocks in the libraries that ship with ExtendSim products. For the ExtendSim CP product you would use the Equation block (Value library) or the Buttons block (Utilities library).

The equation-based blocks provide access to over 1,200 internal functions; you can also use operators to enter logical statements, write compound conditions, and specify loops. The equation can be as simple as performing a mathematical operation on the value from an input connector or it could be as complex as a full programming segment.

You can use include files (See *Include Files* in the Technical Reference) within your equation to reuse your own functions among many equation blocks, and you can use the source code debugger to check

the equation if you have a problem. The equation is automatically compiled when you click OK in the block's dialog.

See the User Reference (How To: Math and Statistics) for detailed information about the equation-based blocks and how to use them.

Using the Python Bridge block

The Python Bridge block extends the modeling capabilities of ExtendSim by enabling communication between ExtendSim models and external Python code. It allows you to pass data between the model and Python, execute Python scripts, and return results to ExtendSim during a simulation run.

The Python Bridge block is useful when you want to incorporate advanced analytics, specialized algorithms, or external data processing that may not be available within standard ExtendSim blocks. For example, you can use Python libraries for statistical analysis, machine learning, data transformation, or integration with external systems.

In a typical workflow, input values from ExtendSim are sent to the Python Bridge block, processed using Python, and the resulting values are returned to the model through the block's output connectors. This enables ExtendSim models to leverage the flexibility and breadth of the Python ecosystem while maintaining the visual modeling approach of block-based simulation.

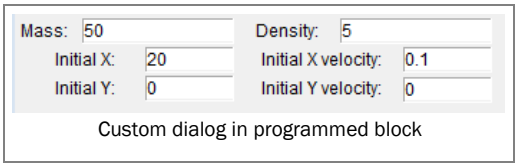
The Python Bridge block complements the Equation block and other tools in ExtendSim. While equation-based blocks are well suited for calculations and logic implemented directly within the ExtendSim environment, the Python Bridge block provides a way to integrate external code and extend model functionality beyond built-in capabilities.

For more information about configuring and using the Python Bridge block, see the User Reference.

Creating custom blocks

You create custom blocks the same way the blocks in the ExtendSim libraries were created—using the ExtendSim IDE and ExtendSim programming tools. And your custom-built blocks can be saved in ExtendSim libraries for use in all models.

The ExtendSim *integrated development environment (IDE)*—the API, source code editor, compiler, dialog editor, graphical UI builder—makes it easy to modify blocks, develop custom features, and create new blocks with custom code, dialogs, and icons.



Custom dialog in programmed block

There are also sophisticated ExtendSim *programming tools* such as:

- Include files
- Conditional compilation
- Diagnostic Log
- Source code debugger
- Block profiler
- Code completion
- Extensions

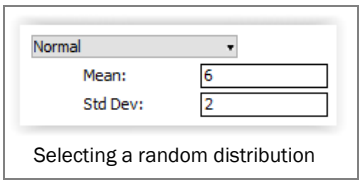
- External source code capabilities

See “Create custom continuous process blocks” on page 18 or the Technical Reference for information and tutorials on creating custom blocks.

Model-building to represent the dynamics of the system

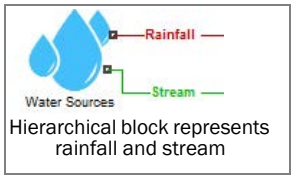
Use these features and capabilities to build a model that represents the dynamic behavior of any continuous process system.

- Point and click model building
  - Use pre-built modeling components from the Value, Chart, and Utilities libraries to represent system behavior. For example, select one of the 35 built-in distributions using the Random Number block (Value library).
  - See “Blocks for building continuous models” on page 17 or the User Reference appendices.



Selecting a random distribution

- Hierarchy
  - Encapsulate sections of a model into hierarchical blocks for scalability.
  - Create generic model structures and store them in libraries for reusability in other models.
  - Customize and animate the icons.

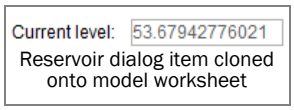


Hierarchical block represents rainfall and stream

Creating a user interface for the model

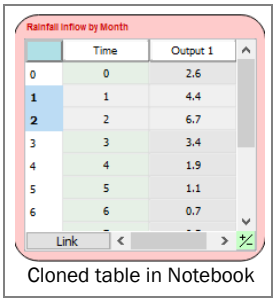
Use these ExtendSim features to create a custom graphical user interface, control panel, or dashboard for the model, simplifying and localizing data entry and displays.

- Clones
  - Exact copies of dialog items, tables, or graphs which behave identically to the original (similar to shortcuts or aliases); no programming required.
  - Place on the model worksheet or notebook to create a custom user interface—a centralized location for making changes or displaying inputs and results.

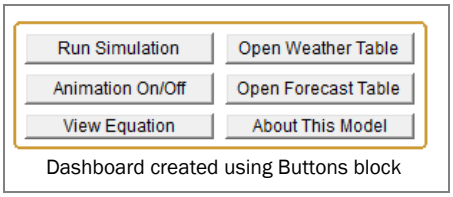


Reservoir dialog item cloned onto model worksheet

- Notebooks
  - Customizable windows for organizing and managing a model’s information—control model parameters, report simulation results, and document the model.
  - Clone objects, insert text and graphics, and paste pictures as shown to the right and on page 11.
- Control panel/dashboard interface
  - Use blocks from the Utilities library to create a dashboard or control panel for the model without programming:

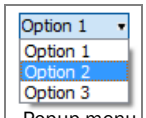


Cloned table in Notebook



Dashboard created using Buttons block

- Buttons block—set up pre-built buttons for a custom interface
- Meter, Slider, and Switch blocks—customizable controls for the model
- Model Interface block—create a button interface for running multiple trials for experimentation
- Popups block—create popup menus with multiple custom options. Can also be used to create buttons for accessing database tables.
- Pause Sim and Run Model blocks—place a button to run or pause the simulation at specified times



Popup menu

  - Create custom buttons using the Buttons block's equation editor or custom interfaces using the ExtendSim IDE or an external application such as VB.net.
- Animation
  - Some Value library blocks, such as the Holding Tank, automatically show 2D animation on their icons if 2D Animation is on during the run.
  - The Animate Value block (Animation 2D-3D library) displays text, a graphic object, or pictures on the worksheet plus can be used to animate a hierarchical block's icon.
  - Use the Equation block (Value library) or the ExtendSim IDE to programmatically create custom animation behavior in blocks you create.



- The Notify block (Value library) plays a sound, stops the simulation, or prompts for an output value when its input is true
- Also see...
  - The ExtendSim database as a tool for presenting data, as discussed on page 47
  - Text, graphic objects, and pictures: ExtendSim supplies tools for, and supports the standard methods of, handling text and graphics. Pictures placed directly on the model worksheet are automatically placed in the background.
  - Interactivity: click a button or change parameters during a simulation run.
  - Named connections: text labels that are used to represent one output at multiple locations in the model, without using connection lines. See the Reservoir models located at Documents/ExtendSim/Examples/Tutorials/Continuous for examples.
  - The Throw Value and Catch Value blocks (Value library): use these blocks to transfer values remotely, even between hierarchical layers, without connection lines.
  - The Navigator tool: this explorer-like window is essential for navigating through the hierarchical structure of a model.

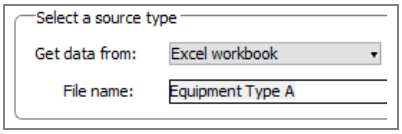


Getting data into the model

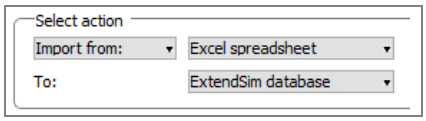
Bring data into the appropriate parts of a model for use before or during the run.

⚠ The preferred method is to get all the data into the model at the beginning of a run; importing data during a run could unnecessarily and dramatically slow the simulation.

- Enter data into model components before the run starts:
  - Type or use Copy and Paste commands or tools to enter data into fields or tables in block dialogs
  - Enter data into parameter fields and tables that have been cloned from block dialogs into notebooks (mentioned in page 45) or onto the worksheet
- Use a Read block (Value library) to get model data, at the start or during a run, from an Excel worksheet, a text file, local table, global array, or ExtendSim internal database.



- Create an ExtendSim database, discussed on page 47 to manage data,; enter inputs directly into its tables
- Import data into a model's internal relational database or (less common) global array; both are discussed on page 47:
  - Data Import Export block—transfer data before, during, or after a run from an external source (Excel, ODBC database, FTP file, text file, ADO database, or XML file) to an internal target (ExtendSim database or global array).

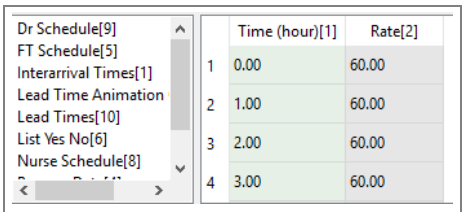


  - Read block, discussed above.
  - Database > Import New Database command—import into a model a database text file that was created in another model or was generated by Excel using the Excel DB Add-In (optional add-on for ExtendSim CP).
- Fit raw data to a distribution with Stat::Fit, as mentioned on page 48

Managing data for use in the model

Localize and manage data so it is available before, during, and after a simulation run.

- Create an ExtendSim internal relational database—a centralized repository for data—and exchange data between the database and the model:
  - Many blocks have a built-in capability to exchange data with the database during the run: Equation, Read, Write, Query Equation, and Data Specs blocks (Value library) and the Popups block (Utilities library)



Database tables; fields for Interarrival Times

  - The Data Import Export block (see) can access data before, during, or after the run
  - Dynamic data linking (DDL) links to data in an ExtendSim database during the run
    - Dynamically link dialog items (parameters and data tables) to internal data structures—internal databases, global arrays, and (programmatically) dynamic arrays.
    - See the Link button on a block's data table or right-click a dialog parameter and choose Create/Edit Dynamic Link



  - Create custom interfaces using the ExtendSim IDE

    - See the *ExtendSim Database Tutorial & Reference*, an eBook located at Documents/ExtendSim/Documentation
- Use the Python Bridge block to extend data management beyond the model using external Python code:
    - Send model data to Python for processing, analysis, or transformation
    - Retrieve results from Python and use them within the simulation
    - Integrate external data sources, analytics workflows, or third-party libraries through Python
    - Use this approach when data handling or processing extends beyond built-in ExtendSim capabilities
- Use another data management method as described more in the Data Management and Exchange chapter of the User Reference:
    - Global array—a two-dimensional array of data accessible by any block in a model.
    - Dynamic array—a multi-dimensional internal data structure only accessible through programming.
    - Linked list—an internal data structure that allows the construction and manipulation of complex lists of data.

Analyzing data and assessing results

Perform calculations, analysis, experimentation, and optimization using information generated by the simulation model.

- Verification and validation
    - Use ExtendSim debugging tools and methods to verify that models operate as expected and to validate that they accurately represent the system. See Debugging Tools in the User Reference.
    - The Model Compare block (Utilities library) helps determine if changes to the model affect results and is helpful when validating models between ExtendSim releases. See Automated test environment in the User Reference.
- Value library blocks:
    - Mean & Variance block—calculate the mean, variation, and standard deviation of the values input during a simulation run; specify the confidence interval.
- Report library blocks
    - Statistics block—accumulate data on certain types of blocks and report statistics in a single table using a statistical method; specify the confidence interval.
    - DB Statistics block—calculate statistics for all the records in a field of an ExtendSim internal database table.
    - Reports Manager block—customize data gathering and filter information by block type or label.
    - Also see the User Reference
- Distribution fitting (Windows only)
    - Determine which distributions, if any, offer a good fit for the underlying raw data.

  - In the Random Number block (Value library) choose Stat::Fit or another distribution fitting software package to perform the analysis. Stat::Fit is an optional add-on for ExtendSim CP.
  - See Stat::Fit in the User Reference.
- Charts and data tables
  - Blocks from the Charts library display a graphical representation of the numbers fed to them as well as a table of the numerical values.
  - For example, see the Line Chart discussed in “On graphs and in data tables” on page 11.
- Evolutionary Optimizer
  - Automatically determine ideal values for parameters, then populate the model with the optimized parameter values.
  - See Optimization in the User Reference.
- Scenario Manager
  - Systematically and strategically explore the outcome of different model configurations and analyze alternatives.
  - Use pre-built experimental designs (DOE) or export to Excel, JMP, or Minitab for further analysis.
  - See Scenario Analysis in the User Reference.
- Sensitivity Analysis
  - Conduct controlled experiments to explore how much of an impact one or more parameters have on model results.
  - See Sensitivity Analysis in the User Reference.
- Also see:
  - Decision, Clear Statistics, and Max & Min blocks in the Value library

Reporting results and exporting data

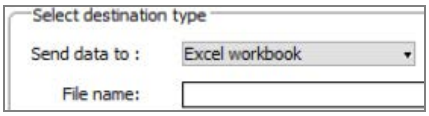
Add blocks to the model to obtain information:

Blocks from the Utilities library:

- Pause Sim block, Pause tool, or Run > Debugging commands pause and step through the simulation to monitor data.
- Add a Record Message block (Utilities library) to the model to capture the messages between connectors

Blocks from the Value library:

- Write block to write model data, during or at the end of a run, to an Excel worksheet, a text file, local table, global array, or an ExtendSim internal database.

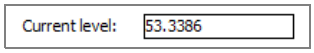


- Notify block (see page 46) sends alerts

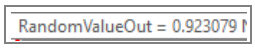
- The Python Bridge block enables communication between ExtendSim and Python. It can exchange data with ExtendSim models and databases, access external data sources, utilize Python libraries, and perform data processing both before and during simulation execution.
- Display Value block displays the input value and can pause between updates
- Blocks from the Report library:
- Reports Manager block, for gathering and reporting filtered data during the simulation run.
- Blocks from the Analysis library:
- Analysis Manager to centralize the management of model parameters and collect model results.
- Monitor and/or export intermediate information and simulation results

⚠ The preferred method is to export the data at the end of a run; exporting during a run could unnecessarily and dramatically slow the simulation.

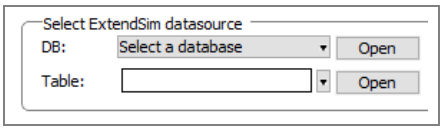
- View data from the following components of your model:
  - Fields and tables in block dialogs such as in the Holding Tank block (Value library)



  - Output parameter fields and tables that have been cloned from block dialogs into notebooks (mentioned on page 45) or onto the worksheet
  - Animation, as mentioned on page 46
  - Tooltips, which display current data each time the cursor is placed over an output connector



- Connect outputs to blocks in the Chart library (mentioned on page 49) to see graphs and tables of data during or after the simulation run
- Create an ExtendSim database as discussed on page 47 to manage data, and see results in its tables or in the DB Line Chart block (Chart library).
- Export data from the model's internal relational database or (less common) from a global array; both are discussed on page 47.
  - Data Import Export block—transfer data during or after a run from an internal source (ExtendSim database or global array) to an external target (Excel, ODBC database, FTP file, text file, ADO database).



  - Write block, discussed above.
  - Database > Export Database command—export an ExtendSim database as a text file so it can be used in another model or can be imported into Excel using the ExtendSim DB Add-In for Excel (optional add-on for ExtendSim CP). To export specific tables, see the Database > Export Selected Tables command.
- Use the Scenario Manager (optional Value library add-on for ExtendSim CP) to export data to Excel, Minitab, or JMP for design of experiments. See page 49.

Communicating with external applications and devices

Methods and technologies for data sharing, communication, and control between ExtendSim and external applications and equipment.

⚠ Some functionality is Windows only

- Pre-built blocks in the Value library:
  - Data Import Export block—exchange data between ExtendSim (internal database or global array) and outside applications and files (Excel, ODBC database, FTP file, text file, ADO databases such as Microsoft Access, SQL Server, and MySQL) at the start, during, or end of a run
  - Read and Write blocks—exchange data between an ExtendSim database, Excel workbook, global array, text file, or local table at the start, during, or end of a run
  - Equation block—see the User Reference.
  - Python Bridge block—enable communication between ExtendSim and Python:
    - •Send model data to Python for processing, analysis, or integration with external tools
    - •Execute Python code during a simulation run and return results to the model
    - •Access Python libraries and external systems through Python scripts
    - •Use this approach when leveraging advanced analytics, machine learning, or external workflows
  - Scenario Manager block (optional add-on for ExtendSim CP)—export scenario results to Excel, JMP, or Minitab for further analysis. Mentioned on page 49.
  - Random Number block—use this block to communicate with JMP, BestFit, ExpertFit, and Stat::Fit to determine the appropriate distribution to use given the raw data
  - Query Equation block see the separate document, ExtendSim Database Tutorial & Reference.
- ExtendSim DB Add-In for Excel (Windows only; optional add-on for ExtendSim CP)
  - Create, edit, and export files between Excel and the ExtendSim internal database
  - Can also use Excel as the master for documenting the data and database structure
  - See the separate document, ExtendSim Database Tutorial & Reference
- ADO (ActiveX Data Objects) for databases (Windows only)
  - Communicate between ExtendSim and external databases—Microsoft Access, SQL Server, and MySQL—using the Data Import Export block (Value library)
  - For programmatic control, the include file “ADO_DBFunctions.h” stores user-defined functions for ADO
  - See the Technical Reference, User-defined functions for ADO
- ExtendSim COM Object Model
  - Primary method for ExtendSim to be controlled and communicated with as an Automation Server

    - Allows external languages to control ExtendSim in many ways, as discussed in "Scripting functions" on page 52
    - See the Technical Reference, ExtendSim as an Automation Server
- ActiveX Automation
    - Primary method for ExtendSim to serve as an Automation Client
    - Access and manipulate (set properties of or call methods on) automation objects that have been developed in and exported by other applications
    - See the Object Mapper block (Utilities library)
    - See the Technical Reference, ExtendSim OLE/COM functions and the section on OLE and ActiveX Automation
- DLL functions
    - Provide custom operations from within a block's code by calling Dynamic Link Libraries (DLLs) of routines or functions written in an external language
    - Primary method for ExtendSim to communicate with DLLs created by other applications or by equipment
    - For many modern use cases, similar functionality can be implemented using the Python Bridge block
    - See the Technical Reference
- Scripting functions
    - Automate the process of building or making changes to models, either from within ExtendSim or by external applications
    - Create custom wizards to simplify tasks or interact with modelers; develop self-modifying models
    - See the Technical Reference, Scripting functions and Scripting techniques
- Internet access
    - Access and manipulate data and files using Internet protocols
    - See the FTP block (ModL Tips library) or program using functions
    - See the Technical Reference, Internet access
- Mailslot functions (Windows only)
    - Communicate and control other computers in a network domain using this messaging functionality supported by the Windows API
    - Send unidirectional messages from a given machine to a specified mailslot on other machines
    - See the Technical Reference, Mailslot
- See also:
    - Command block (Value library)
    - ActiveX Control, DLL Add Block, FTP, ODBC, and Read Device blocks in the ModL Tips library located at Documents/ExtendSim/Libraries/Example Libraries.

- File I/O, Interprocess Communication (IPC), and OLE/COM (Windows only) functions in the Technical Reference

Symbols

+/- button 27

Numerics

2D animation 46

A-C

ActiveX Automation 52
ADO (ActiveX Data Objects) 51
analyzing models 48
Animate Value block (Animation 2D-3D library) 46
animation 2D 46
application areas 2
application window 9
arrays of connectors 15
asterisk (*) 23
Automation 52
backup 23
block
    adding to a model 23
    connecting 26
    connectors 15
    continuous 17
    custom continuous blocks 18
    dialogs 14
    equation-based 43
    Help button 14
    icons 13
    labels 14
    programming 18
    types (mixing) 41
    view 14
bringing data into the model 46
buttons
    for user interface 46
    Run Simulation 10
Buttons block (Utilities library) 46
Catch Value block (Value library) 46
Chart library 18
charts 11
clones 45
Close All Library Windows command 17
COM 52
COM Object Model 51
communicating with external applications 51
Communication between ExtendSim and Python 51
connecting blocks 26
connection lines

Smart Block technique 16
connections
daisy-chaining 42
definition 15
lines 16
methods 16
to multiple value inputs 41
connector
definition 15
Value 41
connectors
arrays of connectors 15
variable connectors 15
continuous process modeling 2
application areas 2
blocks 17
compared to other methodologies 7
equation blocks 17
feedback 38
hierarchy 18
how to build models 17
libraries 17
models 2
programming 18
step size 36
System Dynamics 2
timing 35
where used 3
why simulate 2
control panel 45
creating a user interface 45
Custom order 41
daisy-chaining 42
dashboard 45
data export 50
Data Import Export block (Value library) 47, 50
data input and import 46
data management 47
Data Specs block (Value library) 47
data tables
resizing button 27
database 47
DB Statistics block 48
DDL (dynamic data linking) 47
default view 15
default view for block 14
default view reverse 15
delta time 36

definition 35
determining 36
other than 1 35
setting 36

D-F

dialogs
opening 14
difference equations 2
Differential Equation model 32
differential equations 2
discrete event modeling
continuous blocks in 41
Display Value block (Value library) 50
distribution fitting 48
DLLs 52
double-headed arrow
for expanding a variable connector 15
dragging cursor
definition 15
Drug Ingestion model 32
dt (delta time) 35
stepsize 36
duration of simulation 35
dynamic array 48
Dynamic Link Libraries 52
embedded object 48
End time 22
Equation block (Value library) 17
equation-based blocks 43
Euler integration 38, 39
Evolutionary Optimizer 49
example models 8
experiments
Optimizer 49
Scenario Manager 49
Sensitivity Analysis 49
Export Database command 50
Export Selected Tables command 50
ExtendSim database 47
feedback 38
delays 38
loops 38
File Open command 12
Fish Pond model 32
Flow order 40
Getting Started model 9

G-I

global array 48
global object ID 14
global time units
    setting 22
graphical user interface 45
graphs 11
green +/- resize button 27
GUI 45
Help button 14
hierarchy 18, 45
    Navigator tool 46
Holding Tank block
    integration in 39
    summation in 39
How To
    Analyze and assess 48
    Bring data into the model 46
    Create a user interface 45
    Export data from the model 50
    Manage data for use in the model 47
    Report results 50
    Represent the system 45
icon
    views 15
icon view 14
Import New Database command 47
integrated development environment (IDE) 8, 44
integration
    definition 38
    in the Holding Tank block 39
interactive simulation 14
inter-application communication 51
internal database 47
Internet access 52
interprocess communication 51

J-L

labels for blocks 14
Left to right order 40
library
    library windows 17
    Value library 17
library window 17
linked list 48
mailslot functions 52

managing data 47
Mandelbrot model 32
Markov Chain Weather model 32
Mean & Variance block (Value library) 48
Meter block (Utilities library) 46

M-O

model
  adding blocks 23
  backup file 23
  basics 12
  block definition 13
  connection lines 16
  connectors 15
  continuous 2
  differential equation 36
  example 8
  icons 13
  integration methods 38, 39
  interactive 14
  opening 12
  order of calculation for blocks 40
  parts of a model 12
  running 10
  validation 48
  verification 48
Model Compare block 48
Model Interface block (Utilities library) 46
modeling
  continuous 2
models
  Quick Start 9
Monte Carlo model 32
multi-threaded run mode 30
multi-threading capability 30
names of blocks 14
Navigator tool 46
Noisy FM System model 32
notebooks 45
Notify block (Value library) 46
object ID 14
OLE/COM 52
Open All Library Windows command 17
opening a model 12
optimization 49
order
  of simulation 40
Pause Sim block (Utilities library) 46
Planet Dance model 32

plotters 11
Point and Click technique 24
Popups block (Utilities library) 46, 47

P-R

Predator Prey model 32
programming tools 44
Python Bridge block 44
Python Bridge block to extend data mgmt. 48
Query Equation block (Value library) 47
Quick Start model 9
Random Number block 49
Random Number block (Value library) 45
Read block (Value library) 47
remote transfer of values 46
Report library 18
representing the system 45
Reservoir 2 model 31
Reservoir 3 model 32
Reservoir model 11, 21
resize button 27
results (displaying on charts) 11
Run Model block (Utilities library) 46
run modes
    Fastest 30
    multi-threaded 30
    single-threaded 30
run parameters 22
Run Simulation button 10
Run Simulation command 10
running a model 10
Scenario Manager 49
scripting functions 52
sending data out of the model 50
Sensitivity Analysis 49
Setup tab 22
simulation
    continuous modeling 2
    Custom order 41
    duration 35
    Flow order 40
    integration methods 38
    interactive 14
    Left to right order 40
    order 40
    parameters 22

results on graphs 11
running 10
runs 22
time 35
simulation order 40
Simulation Setup command 22, 35
single-threaded run mode 30
Slider block (Utilities library) 46
Smart Block technique 16, 24
Smart Connection technique 26
StatFit 48
Statistics blocks (Value library) 48
steps
number of 36
size 36
storing data 47

S-U

Switch block (Utilities library) 46
System Dynamics 2
system representation 45
systems
continuous 2
Throw Value block (Value library) 46
time units
global 22
tooltip 13
tooltips 50
tutorial
simple model 21
user interface 45
buttons 46
creating 45
validation 48
Value connectors 41
Value library 17
variable connectors 15
verification 48
views 14

V-Z

ways to make connections 16
wizards 52
Write block (Value library) 50