

Vensim[®] PLE

Vensim[®] PLE Plus

Personal Learning Edition

with
Causal Tracing[®]
and
Reality Check[®]

User's Guide Version 4

Information in this document is subject to change without notice and does not constitute a representation or warranty on the part of Ventana Systems, Inc. The software described in this document is furnished under a license agreement and may be used or copied only in accordance with the terms of that agreement.

© Copyright 1988-1999 Ventana Systems, Inc.

All Rights Reserved except the Following Reproduction Rights are Granted:

This documentation may be reproduced by individuals for their own use and by schools and accredited academic institutions for instructional and research use. Any other reproduction is prohibited without the express written consent of Ventana Systems, Inc. This page must be included in any reproduction of any part of the Vensim PLE User's guide.

The Vensim PLE Software may be reproduced and distributed free of charge as a bound, self installing executable. Any sale of Vensim PLE is prohibited without the express written consent of Ventana Systems, Inc. Any distribution of the files installed in support of Vensim PLE is prohibited without the express written consent of Ventana Systems, Inc.

The Vensim PLE Plus software must be licensed from Ventana or its authorized agent.

Patent and Trademark Notices

Causal Tracing, Reality Check, Vensim and Ventana are registered trademarks of Ventana Systems, Inc. Venapp and the Ventana Logo are trademarks of Ventana Systems, Inc. Ventana is a registered service mark of Ventana Systems, Inc. Vensim is covered by United States Patents 5,428,740 and 5,446,652. Patents are pending in the United States and other countries related to portions of the Vensim software.

Other brand or product names are trademarks or registered trademarks of their respective owners.

Printed in the United States of America.

Table of Contents

1 Introduction	1
About Vensim PLE	1
About this Manual	1
<i>How this Manual is Organized, Style Conventions, Important Notes, About Directories, About Screen Shots, About the Mouse, About Tab Dialog Boxes</i>	
Installation	5
<i>Start the Installation Program, License Agreement, Registration Code, Installation Directory, Other Vensim Configurations</i>	

2 The Vensim User Interface	7
Main Features	7
Title Bar	8
Menu	8
Toolbar	8
The Window Classes	9
<i>Moving Among Window Classes, Moving Among Windows within a Class</i>	
The Build Window	10
<i>Sketch Tools, Status Bar</i>	
Output Windows	12
<i>Analysis Tools, Analysis Tool Output</i>	
The Control Panel	14

3 A Hands-On Example	15
Modeling with Vensim 15	
<i>The Workforce Inventory Example</i>	15
Starting Vensim 16	
<i>Opening the Model</i>	
Examining Structure 17	
Simulating the Model 18	
Examining Behavior 19	
Causal Tracing 21	
Simulation Experiments 24	
Making a Custom Graph 25	

4 Causal Loop Diagrams	29
Vensim Models 29	
<i>Drawing Sketches, Mouse Tips</i>	
Constructing a Causal Loop Diagram 30	
<i>Project Model (project.mdl)</i>	
Modifying diagrams 34	
<i>Sketch Layout, Sketch Options, Adding Comments and Graphics, Refining the Model</i>	
Printing and Exporting the Sketch 40	
Structural Analysis of Diagrams 41	
<i>Analysis Tools, Causal Tracing with Trees</i>	

5 Stock and Flow Diagrams	45
Building a diagram (customer.mdl) 45	
<i>Entering Levels, Creating Rates, Bending Rate Pipes, Adding Auxiliaries and Arrows, More Structure</i>	
Customizing Diagrams 50	
<i>Sketch Options, Variable Shapes</i>	

6 Building a Simulation Model	53
A Population Model 53	
Vensim Conventions 53	
Sketching the Rabbit Model (pop.mdl) 54	
Writing Equations 55	
Checking for Model Syntax and Units Errors 58	
<i>Units Equivalents (Synonyms)</i>	
Simulating the Model 59	
Model Analysis 59	
Comparing Simulations 60	
<i>Exponential Growth, Setting Up a Simulation Experiment, Causes Strip Graph, Runs Compare, Exponential Decay, Input and Output Objects</i>	

7 Building a Function with Lookups	65
Limits to Rabbit Growth 65	
Normalized Lookups 66	
<i>Sketching the Model (rabbit.mdl), Entering Equations</i>	
Creating and Normalizing Lookups 69	
<i>Editing Values</i>	
Checking for Model Syntax and Units Errors 72	
Simulating the Model 72	
Model Analysis 72	
Separate Normalized Variables 73	
<i>Simulation, Changing Model Lookups</i>	
Named Lookups 75	
8 Multiple Views	77
How Views Work 77	
Customer Diffusion Model 77	
<i>Building the Diffusion Model (cust1.mdl), Simulating the Model, Model Analysis, Naming and Saving Your Model</i>	
Adding the Capacity View (cust2.mdl) 80	
<i>Shadow Variable Tool, Adding Equations, Altering An Equation, Adding a Sales Revenue View, More Equations, Simulating the Model, Analyzing the Model, Saving Your Model</i>	
Detailed Capacity Model (cust3.mdl) 86	
<i>Copying and Pasting, Capacity View Equations, Units Synonyms</i>	
Simulating and Analyzing the Model 89	
<i>Capacity Investment Policy</i>	
9 Customizing Output	91
Output from Graph Analysis Tools 91	
Custom Graphs 92	
10 Games In Vensim	95
What Are Games? 95	
The Real Estate Game (houses.mdl) 95	
<i>Model Structure, Built-in Functions, Adding Game Variables, Simulating the Model</i>	
Playing the Game 100	
<i>Moving Forward in a Game, Backing Up in a Game</i>	
11 Input Output Controls	103
Word of Mouth Sales 103	
<i>wom1.mdl Equations</i>	
Output Controls 105	
Input Controls 107	
Gaming Control 109	
Publishing the Model 111	
<i>Game Interval, Commentary and Navigation Links, Test It Out, Binary Format Save</i>	

12 Sensitivity Testing	115
Monte Carlo Simulations 115	
Market Growth Model (sales.mdl) 115	
<i>Sales.mdl Equations 116, Base Simulation 117</i>	
Uncertainty in Multiple Parameters 118	
<i>Random Uniform Distribution, Random Normal Distribution</i>	
Save Lists 121	
Sensitivity Simulations 122	
Time Graph Sensitivity Output 123	
<i>Confidence Bounds</i>	

13 Using Data in Models	125
Types of Data Use 125	
Using Data to Drive a Model (cfc.mdl) 125	
Data Variable with Data Function 127	
<i>Simulating</i>	
Data Variable with Imported Data 128	
Importing Text-Formatted Data (.dat) 129	
<i>Simulation</i>	
Importing Spreadsheet Data 132	
<i>Simulation</i>	

14 Physical Systems	135
Exponential Growth — Snowball (snowball.mdl) 135	
Exponential Decay — Coffee Cooling (cool.mdl) 136	
Speed and Acceleration (speed.mdl) 137	
Gravitational Attraction (gravity.mdl) 138	
CFCs in the Atmosphere (cfc.mdl) 140	

15 Biological Systems	143
Easter Island Trees (eit.mdl) 143	
Addiction — Caffeine in the Body (caffeine.mdl) 145	
Rabbit Ecology (rabbit.mdl) 147	
Rabbit and Fox Ecology (rabfox.mdl) 149	
<i>Simulation, Analysis, Experiments</i>	

16 Workforce, Inventory and Oscillation	153
Background 153	
<i>Reference Modes</i> 153	
Dynamic Hypothesis 154	
Workforce / Inventory Model (wfinv1.mdl) 154	
<i>Workforce 155, Behavioral Relationships</i>	
Equation Set wfinv1.mdl 157	
Analysis 158	
Model Refinement (wfinv2.mdl) 159	
Additional Equations 160	
Refined Model Behavior 160	
Phasing and Oscillation 162	
<i>Sensitivity</i>	
17 Pendulums and Oscillation	165
The Pendulum (pendulum.mdl) 165	
Simulating the Pendulum 167	
<i>Euler Integration, Runge-Kutta Integration</i>	
The Damped Pendulum 169	
Conclusions on Integration Techniques 170	
18 Reality Check	171
Models and Reality 171	
<i>Domains of Expertise</i> 172	
Defining Reality Check Equations 172	
<i>Test Inputs, Constraints</i>	
Simulation and Reality Check 177	
<i>Active Constraint Checking, Passive Constraint Checking, Error Reporting</i>	
Entering Reality Check Equations 178	
<i>Equation Editor</i>	
Running Reality Check 180	
<i>Reality Check Results, Reviewing Simulation Results</i>	
Reality Check and Yeast Growth 183	
<i>Test Input and Constraint Equations, An Initial Model, Temperature, Divisions and Terminations,</i>	
<i>Divisions as influenced by Water and Sugar, Water and Sugar as Influenced by Divisions,</i>	
<i>Conclusion</i>	
19 Functions and Keywords	193
Summary List of Functions 193	
Detailed Function Descriptions 195	
Lookups 214	
<i>Using Lookups</i>	

Appendix A - Models that Come with Vensim PLE	217
Chapter Models 217	
Sample Models 217	

Appendix B - Information Resources	219
Books 219	
<i>System Dynamics, Other Areas</i>	
Roadmaps 220	
Publishers 220	
System Dynamics Society 221	
Internet Resources 221	
<i>World Wide Web, Mailing Lists</i>	

License and Support	223
Vensim PLE Software License 223	
Vensim PLE Plus Software License 225	
Vensim PLE Support 226	

About Vensim PLE

Vensim is a visual modeling tool that allows you to conceptualize, document, simulate, analyze and optimize models of dynamic systems. Vensim PLE (for Personal Learning Edition) and PLE Plus are configurations of Vensim that are designed to make it easier to learn system dynamics. Vensim PLE provides a simple and flexible way of building simulation models from causal loop or stock and flow diagrams. Vensim PLE Plus features almost the same interface but has additional functionality that increases the scope of things you can do.

By connecting words with arrows, relationships among system variables are entered and recorded as causal connections. This information is used by the Equation Editor to help you form a complete simulation model. You can analyze your model throughout the building process, look at the causes and uses of a variable and the feedback loops involving the variable. When you have built a model that can be simulated, Vensim lets you thoroughly explore the behavior of the model.

Vensim PLE and PLE Plus have been designed for both personal learning and formal education, and are appropriate for a first course in system dynamics at high school, undergraduate, and graduate levels. Although there are no limitations on model size, Vensim PLE is ideally suited to building small system dynamics model. Vensim PLE Plus, because it supports multiple views, will work quite well with larger models. For complex commercial applications, higher level graduate courses, research and advanced undergraduate work, another Vensim configuration might be appropriate. If you want to learn about other Vensim configurations, please call, or visit our web site at <http://www.vensim.com>.

About this Manual

This Manual will guide you through the main features of the Vensim simulation software, introducing Vensim in a hands-on environment where you can examine existing models, and construct your own causal loop diagrams, stock and flow diagrams, and simulation models. Most of the models in this Manual are presented with all the structure and all the equations you need so that you can build the models yourself.

You will get the most out of this manual if you work through the examples in sequence, at least through Chapter 9. Getting good at building models requires lots of practice, and working through this Manual can provide some of that practice.

How this Manual is Organized

This Manual is broadly divided in five parts—overview, tutorial, topics, modeling examples and reference. The first nine chapters provide an overview and cover the mechanics of building models with Vensim: how to draw diagrams, add equations, simulate and analyze models, and display output. Chapters 10 through 13 demonstrate more advanced operations available in Vensim PLE Plus using

1: Vensim PLE User's Guide

existing sample models (which you can also build). Chapters 14 through 17 provide examples of different types of models. Finally Chapters 18 describes the Reality Check functionality in Vensim and Chapter 19 provides reference material on functions available in Vensim PLE and PLE Plus.

Overview

Chapter 1 provides an overview of this Manual and of Vensim.

Chapter 2 introduces you to the Vensim User Interface. This chapter provides an overview of Vensim, along with information on the Sketch tools, Analysis tools, and Control windows.

Tutorial

Chapter 3 provides hands-on experience in simulating and analyzing an existing model.

Chapter 4 introduces you to the construction and use of causal loop diagrams. Structural analysis of diagrams using Analysis tools is also described.

Chapter 5 covers building stock and flow (Level and Rate) diagrams.

Chapter 6 steps you through the construction of a simulation model of the growth of a population. This problem helps you learn the mechanics of building, simulating, and analyzing models with Vensim.

Chapter 7 describes how to create and use Lookups (arbitrary functions).

Chapter 8 develops a model with multiple views (PLE Plus), allowing you to split a model into different sectors. Alternate directions are included for working in Vensim PLE.

Chapter 9 shows you how to customize output graphs from the Analysis tools. The Custom Graph editor, in which you can create customized graphical output of multiple variables is also described.

Topics (PLE Plus Only)

Chapter 10 explains how to use models as games, or "flight simulators", where you can step progressively through time while making decisions on the way.

Chapter 11 shows how to make use of Input Output Controls and Navigation Links to make a model easier and more fun to use and present to others.

Chapter 12 provides an example of Monte Carlo sensitivity testing. You set parameters with uncertainty values and then run sensitivity analysis to determine uncertainties in particular variables over the simulation time.

Chapter 13 describes how to use data in models. Data variables are defined which access exogenous time series and drive model behavior. This chapter covers importing data from text files and from spreadsheets.

Modeling Examples

Chapter 14 introduces physical systems.

Chapter 15 describes biological systems.

Chapter 16 works through the Workforce-Inventory oscillation model.

Chapter 17 investigates oscillations and integration techniques.

Reference

Chapter 18 covers the Reality Check feature allowing model validity testing.

Chapter 19 gives a list of the built-in functions available in PLE and PLE Plus.

Style Conventions

To differentiate among the many elements in Vensim, this Manual follows some style conventions:

- Names of files stored on disk and their extensions are shown in *italic* (e.g. *project.mdl*). The names of datasets are shown in *italic* without the *.vdf* extension (e.g. *baserun*).
- Names of variables and equations in a Vensim model are in *Italic Courier font*, (e.g., *Population*).
- Names of Vensim menu items, controls, buttons, tools, toolbars, and names in dialog boxes are all capitalized, (e.g., Control Panel), and usually appear in **bold font** if the name refers to an object in Vensim that you will be selecting or acting on (e.g., press the **Simulate** button).
- Actions that you need to perform use a triangular bullet (e.g.,
 - ▶ Click the **Open Model** button and choose the model *wfinv.mdl*).

Important Notes

About Directories

All of the models discussed in this Manual are contained in the *plemodel* subdirectory of the directory in which Vensim is installed. Under Windows 95 and beyond this will typically be *c:\Program Files\Vensim*. Under Windows 3.1 it will normally be *c:\vensim*. On the Macintosh it will normally be *Vensim* on the Desktop. However, you can install Vensim in any location you choose so we will name directories using the path starting with *plemodel* as in *plemodel\chap07\complete*. On the Macintosh this just means the *complete* folder in the *chap07* folder in the *plemodel* folder.

When you are working with your own models we strongly recommend that you store them in a different directory that is not a subdirectory of Vensim. For the purposes of this Manual saving your work in subdirectories of the *plemodel* directory is perfectly reasonable.

About Screen Shots

There are very slight differences between the appearance of Vensim PLE and Vensim PLE Plus, essentially a few more Tools are available in PLE Plus. Most of the pictures of screens in this Manual were taken from Vensim PLE Plus. If you see a Tool that does not appear in your copy of Vensim it is not a problem.

About the Mouse

Windows computers have left and right mouse buttons, but Macintosh computers have only one mouse button. Vensim makes use of the left and right buttons on PCs as described below. Macintosh users will need to use their mouse button and the Ctrl key (for right button clicks) as described below.

Left Button

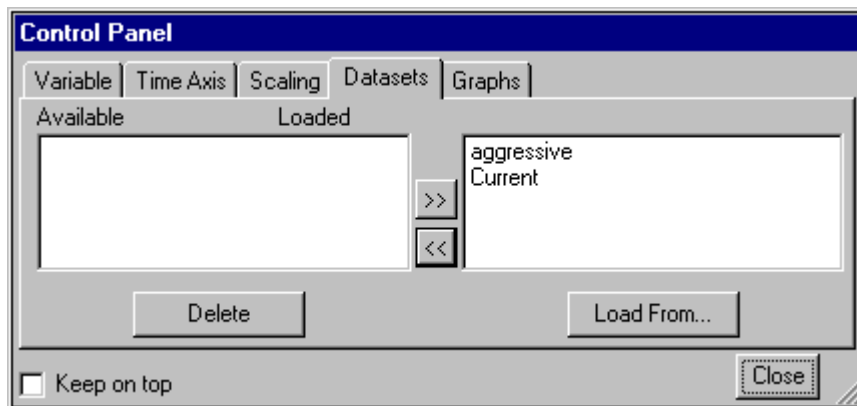
The left button is used to perform almost all operations in Vensim, such as selecting a menu, clicking a button, dragging graphs or sketch objects. Whenever the Manual requires a mouse click without mention of left or right, perform a click with the left button. Macintosh users should click the only button.

Right Button

The right button is used to set options for Sketch tools, Analysis tools, and Sketch objects. When a right button click is required, click the right button on PCs. Macintosh users need to click the mouse button while holding down the Ctrl key or the Apple key (Ctrl + Click or ⌘ + Click).

About Tab Dialog Boxes

Tab Dialogs are special dialog boxes common in Windows 95 and later. These dialog boxes simplify controls by separating information into different "folders" with tabs. You can then switch between folders by clicking on the appropriate tab. Examples of tab dialog boxes are the Simulation Control, the Equation Editor, and the Control Panel (shown below):



In this picture, the **Datasets** tab has been selected showing two loaded simulation runs. The **Variable**, **Time Axis**, **Scaling**, and **Graph** controls can be selected by clicking on the tab desired.

Installation

The exact sequence and appearance of dialogs for installation will vary depending on both the target platform and the type of media (download, CD, disk) on which you got the installation program. The basic steps for installing are, however, the same.

Start the Installation Program

If you downloaded Vensim from our Web site the installation program will have a name such as *venplp32.exe* and be located in the directory or folder to which you downloaded it. You choose this directory when your web browser queries you for the location you want to save the downloaded file to. Normally your web browser also give you the option to rename this file and so you should remember where you saved it. If you are on the Macintosh download files come across as binhex files and you may need to convert this to an application (there are a number of programs that do this and some web browsers do this automatically).

If you have gotten the installation program on a CD or floppy disk look for an executable file such as *setup.exe* or, on the Mac, *Vensim PLE Installer*. Again the installation program may have been renamed so you may need to look for other names.

If you are starting the installer from a CD it may start itself when you insert the CD. Otherwise you will need to start the installation program by double clicking on it typing its name at a DOS prompt or choosing it from the Start>Run... dialog.

The installation program may ask you what you want to install. Vensim PLE Plus and Vensim PLE for commercial use require a Registration Code. Vensim PLE for educational or evaluation use do not require a registration code. If you do not have a Registration Code you will need to Install Vensim PLE for educational or evaluation use.

License Agreement

Before you can install Vensim you will need to agree to the terms of a license agreement. For your convenience this licensee agreement is repeated at the end of this Manual. If you agree to the terms of the license agreement indicate this and continue with the installation. If you do not agree to the terms of the license agreement you may return the software for a refund of whatever license fees you have paid.

Registration Code

If you are installing Vensim PLE Plus, or Vensim PLE for commercial use, you will receive a Registration Code when you purchase the software license. The Registration Code is a series of letters, digits and dashes. If you purchased your license on-line, your Registration Code will be sent to you via email, otherwise it will be printed on a label placed on the inside cover of your manual.

Enter the Registration Code exactly as it appears including capitalization and dashes. If you received the code via email, it is easiest to copy it from the email and paste it in. If you make a mistake entering the code, you will be asked to edit the code you entered. Double check to be sure it is identical with the one you received.

Installation Directory

You can choose the directory or folder into which you want to install Vensim. By default this will be *c:\Program Files\Vensim* (*c:\Vensim* under Win 3.1 and *Vensim* on the Desktop on the Mac. However, you can choose to install Vensim anywhere that you want. When we refer to directories in this Manual they are subdirectories of the directory in which Vensim has been installed.

Other Vensim Configurations

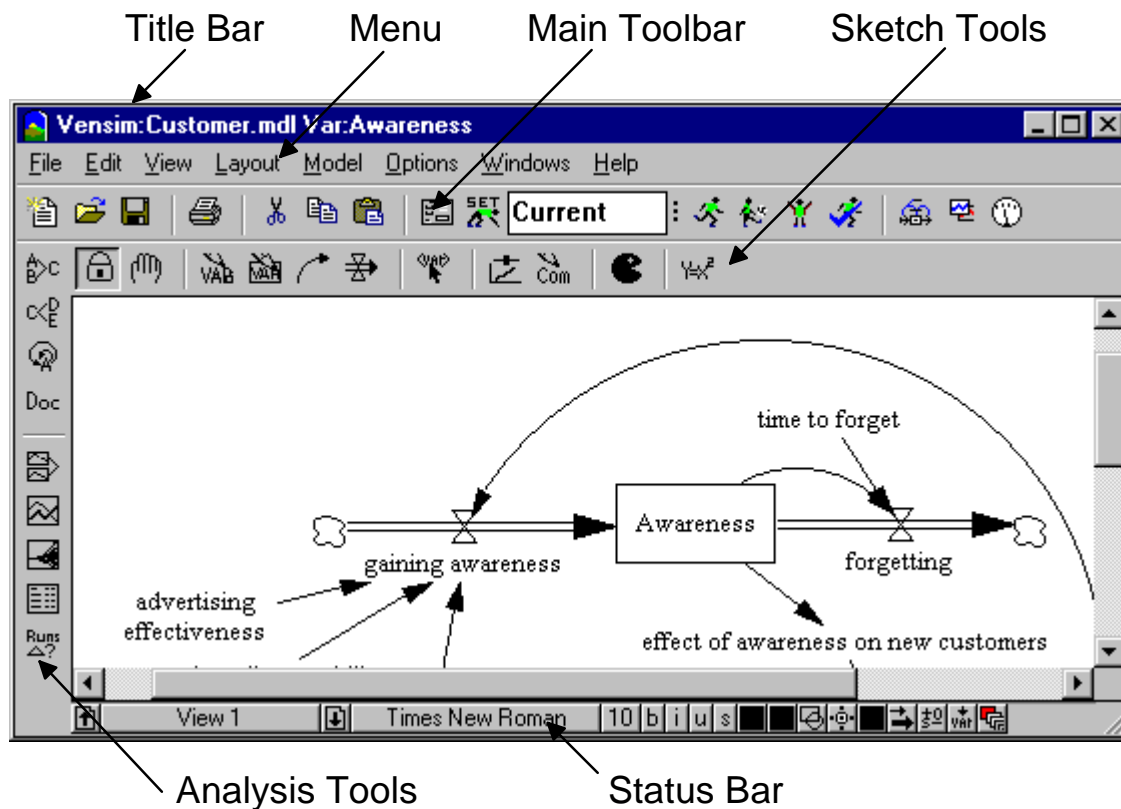
Version 4 will install Vensim PLE as an executable named *venple.exe* (*Vensim PLE* on the Macintosh) and Vensim PLE Plus as an executable named *venplep.exe* (*Vensim PLE Plus* on the Macintosh). Therefore, you can install Vensim PLE or PLE Plus and another Vensim configuration in the same directory without any conflicts. Vensim PLE and PLE Plus ship with identical models and these will be installed into the *plemodel* subdirectory. Other Vensim configurations install models into a *models* subdirectory so, again, there will be no conflict. Vensim PLE and PLE Plus also store a limited amount of configuration in files called, respectively, *venple.ini* and *venplep.ini*.

The long and short of this is that installing Vensim PLE along with another Vensim configuration in the same place will not cause any problems.

2 The Vensim User Interface

Main Features

Vensim uses an interface that resembles a workbench and a set of tools. The main Vensim window is the Workbench, which always includes the Title Bar, the Menu, the Toolbar, and the Analysis tools. When Vensim has a model open (as shown below), the Sketch tools and the Status Bar also appear.



Title Bar

The Title Bar shows two important items: the model that is open (e.g., *Sales.mdl*) and the Workbench Variable (e.g., *sales force productivity*).



The Workbench Variable is any variable in the model that you have selected and for which you want more information, such as the dynamic behavior of that variable. The Workbench Variable is selected by double-clicking on a variable or by using the Variable Selection Control in the Control Panel ("Control Panel" later in this Chapter).

Menu

All operations in Vensim can be performed from the menu, with the exception of using Sketch tools and Analysis tools.

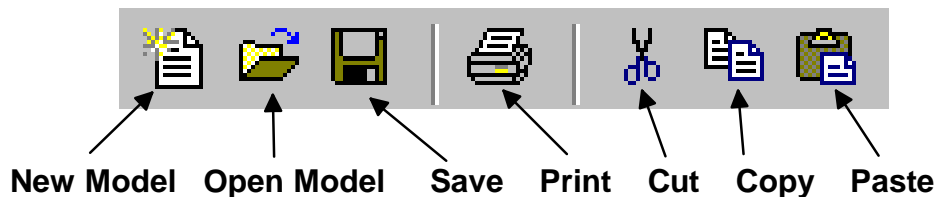


- The **File** menu contains common functions such as Open Model, Save, Print, etc.
- The **Edit** menu allows you to copy and paste selected portions of your model. You can also search for a variable in your model.
- The **View** menu has options for manipulating the sketch of the model and for viewing a model as text-only (available only in Vensim Professional and DSS).
- The **Layout** menu allows you to manipulate the position and size of elements in the sketch.
- The **Model** menu provides access to the Simulation Control and the Time Bounds dialogs, the model checking features, and importing and exporting datasets.
- The **Options** menu sets Vensim's global options.
- The **Windows** menu enables you to switch among different open windows.
- The **Help** menu provides access to the on-line help system.

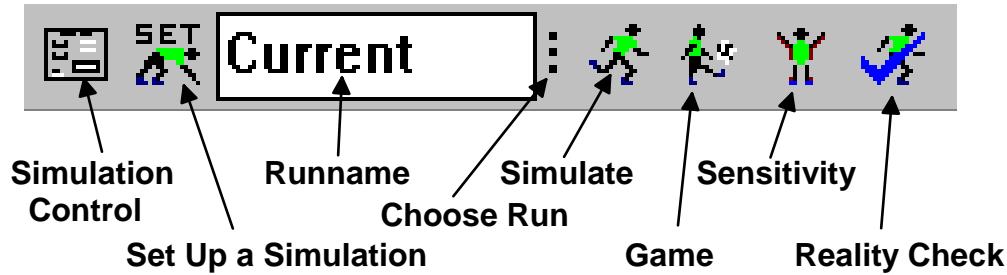
Menus are window-sensitive. The menu acts on whichever window currently is active.

Toolbar

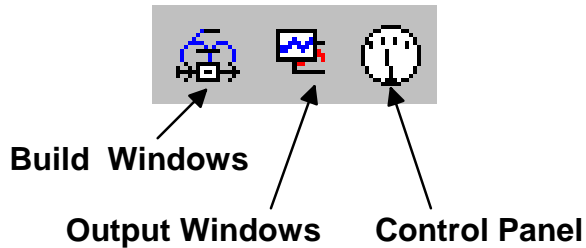
The Toolbar provides buttons for some of the most commonly used menu items and simulation features. The first set of buttons access some File and Edit menu items.



The next several buttons and the **Runname** editing box are used for simulating models.



The last few buttons access the window classes.



The Window Classes

Vensim contains several types or classes of windows:

1. The Build Window is used for constructing new models, or for modifying, navigating, and simulating existing models. Only one Build window can be open at a time.
2. Output Windows are created by Vensim's Analysis tools, and include graphs, tables and lists.
3. The Control Window is the Control Panel, a tab dialog box used to control Vensim's internal settings..

Moving Among Window Classes

When a window is first selected or created, that window moves to the top and is active while all other windows become inactive. You can only work in an active window. Four different methods allow you to move between the window classes:

1. Click on the appropriate window button on the Toolbar.
2. Press Ctrl + Shift + Tab to cycle among the window classes.
3. From the **Windows** menu, select **Pop Build Forward**, **Pop Output Forward** or **Control Panel**.
4. Use the mouse cursor and click on the appropriate window, if the window is visible.

2: Vensim PLE User's Guide

The last method works particularly well for the Build Window, which is the largest and is not usually covered when other windows are active.

Moving Among Windows within a Class

Four methods enable you to cycle through the open windows within a class:

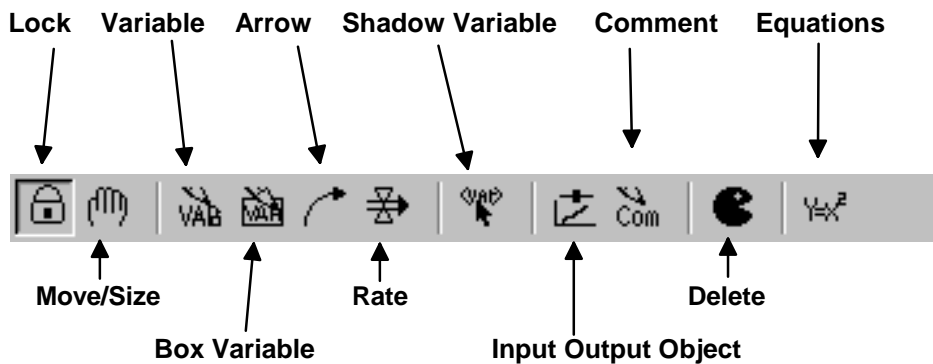
1. Click on the window class button.
2. Press Ctrl + Tab (or Shift + Tab to move in the opposite direction).
3. From the **Windows** menu, select **Output Window List** for the Output windows.
4. Use the mouse and click on the appropriate window, if the window is visible.

The Build Window

The Build Window is used to create models in Vensim by sketching the structure of the model and for writing equations. The Status Bar provides buttons for modifying the sketch. In PLE Plus models can be built from several different sketches or sketch views. Each sketch view shows a part of the model, much like each page in a book tells part of a story.

Sketch Tools

Sketch tools are grouped into a Sketch toolset that appears at the top. This toolset contains the Sketch tools needed for building models. It is not possible to configure Sketch tools or modify the Sketch toolset in Vensim PLE and PLE Plus.



The Sketch tools are:

- **Lock** — sketch is locked. Pointer can select sketch objects and the Workbench Variable but cannot move sketch objects.
- **Move/Size** — move, sizes and selects sketch objects: variables, arrows, etc.
- **Variable** — creates variables (i.e., Constants, Auxiliaries, etc.).
- **Box Variable** — create variables with a box shape (used for Levels or Stocks).
- **Arrow** — creates straight or curved arrows.
- **Rate** — creates Rate (or flow) constructs, consisting of perpendicular arrows, a valve and, if necessary, sources and sinks (clouds).
- **Shadow Variable** — adds an existing model variable to the sketch view as a shadow variable (without adding its causes).
- **Input Output Object** — add input Sliders and output graphs and tables to the sketch (PLE Plus only).
- **Sketch Comment** — adds comments and pictures to the sketch.
- **Delete** — deletes structure, variables in the model, and comments in a sketch.
- **Equations** — creates and edits model equations using the Equation Editor.

To build a model, first select a Sketch tool by clicking on it with the mouse. You can also select a tool by pressing a character on the regular keyboard (*not* the numeric keypad). Use 1 for the first tool, 2 for the second and so on (0 is the 10th, Q the 11th). Note that this only works when the Build window is active.

Move the mouse to the sketch view and click once with the left mouse button to apply the tool (for Arrows and Rates, first click once, then move the mouse and click once again). The Sketch tool you chose stays active until you choose another — just keep applying it to the sketch.

Status Bar

The Status Bar shows the state of the sketch and objects in the sketch. The Status Bar contains buttons for changing the state of selected objects, and moving to another View.



A number of sketch attributes can be controlled from the Status Bar, including:

- Change characteristics on selected variables; font type, size, bold, italic, underline, strikethrough.
- Variable color, box color, surround shape, text position, arrow color, arrow width, arrow polarity etc.

When using the Text Editor (Vensim Professional and DSS), the Status Bar changes to reflect text editing operations.

Output Windows

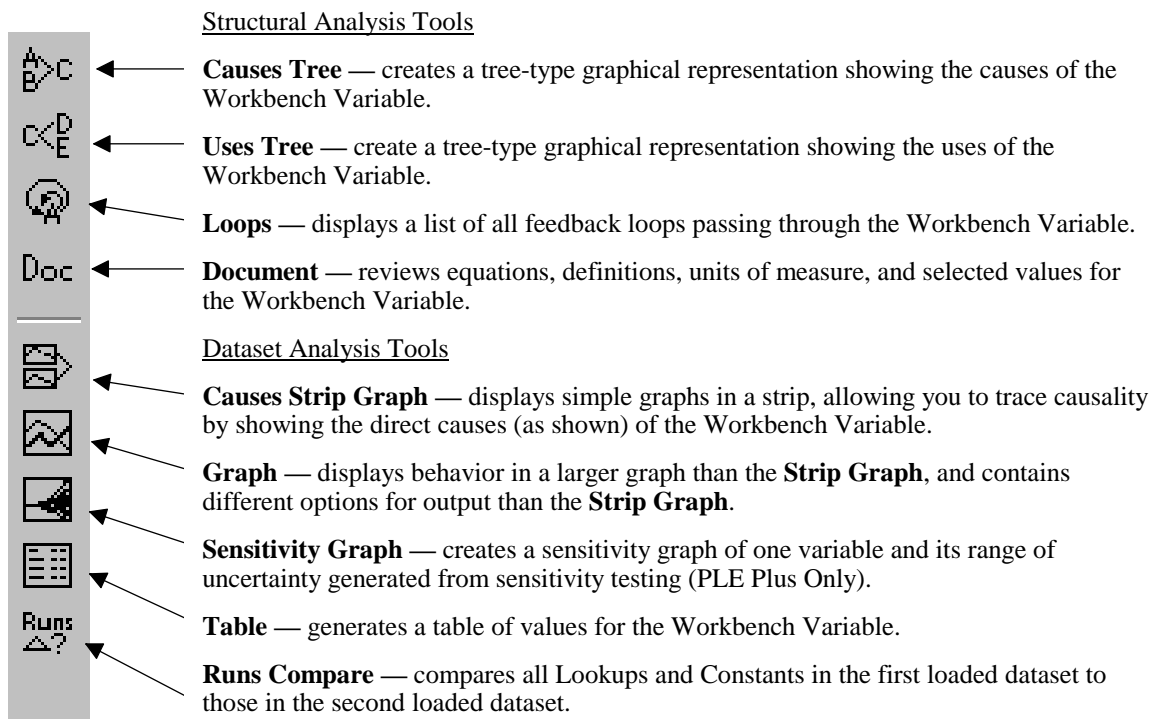
Output Windows are generated by clicking on an Analysis tool. The Analysis tools gather information from the model and display the information in a window as a diagram, graph, or text, depending on the particular tool. Dozens of these windows can be open simultaneously, and a particular window can be closed individually by clicking the **Close** button in the top left or top right corner, or all windows can be closed at once using the menu item **Windows>Close All Output**.

Analysis Tools

Analysis tools are grouped into the Analysis toolset. The Analysis toolset contains the Analysis tools needed for investigating models. The Analysis tools are used to show information about the Workbench Variable, either its place or value in the model, or its behavior from simulation datasets.

Analysis tools can not be configured and the Analysis toolset can not be modified in Vensim PLE and Vensim PLE Plus.

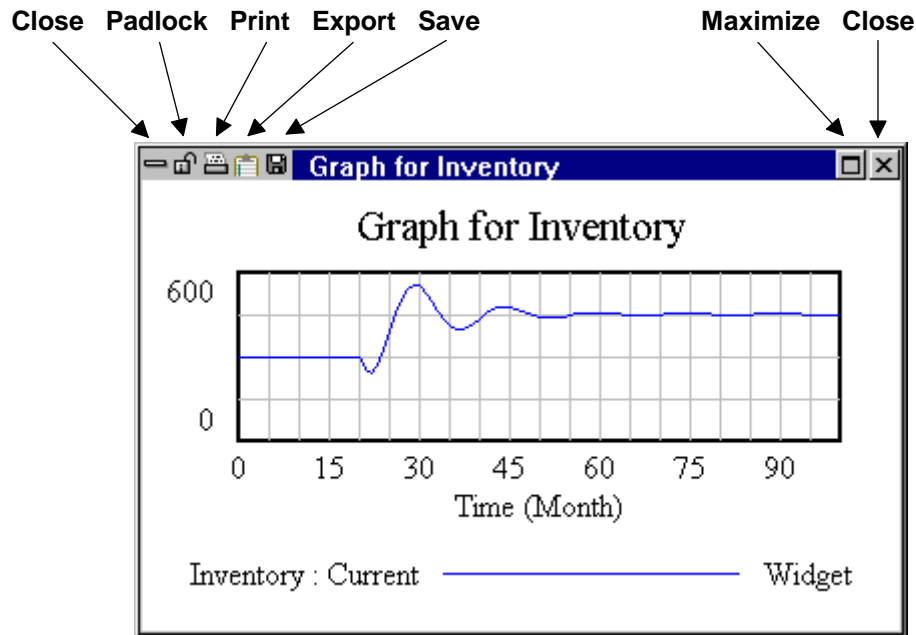
Shown below is the Analysis toolset



Analysis Tool Output

Clicking on an Analysis tool generates a new window with formatted output, except for the Table and Document tools, which add information to any existing Table or Document Output window. The output of a tool remains on screen until you remove it, and *is not* updated as changes are made to the model.

An example of Analysis tool output is the graph displayed below. Descriptions of buttons common to all Output windows follows.

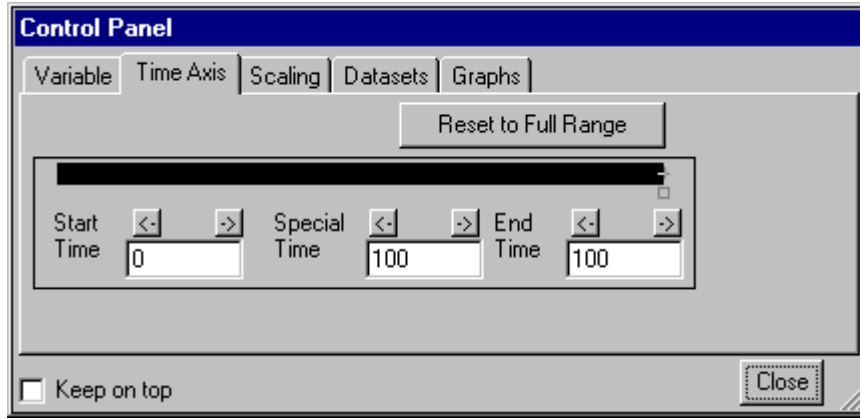


- If you change a model or make a new dataset, you can delete the old output easily and quickly by clicking one of the **Close** buttons located in the top left or top right corner or pressing the **Del** key.
- You can delete all Output windows by selecting the menu item **Windows>Close All Output**.
- You can prevent an Output window from being closed by clicking on the **Padlock** button in the top left-hand corner to lock the window. Clicking on the **Padlock** again will unlock the window.
- You can permanently save information in an Output window by either clicking the **Save** button (to save to a file) or the **Export** button (to save to the clipboard for pasting into another application) while the Output window is active.
- If you remove the output, you can reproduce it easily by invoking the tool that generated it again (unless you have changed things in the model or set special Constant or Lookup table values).

Analysis tool output is easy to create and easy to get rid of. Analysis tools do not create information, but put existing information into a more useful and digestible form.

The Control Panel

The Control Panel allows you to change internal settings that govern the operation of Vensim, such as which Workbench Variable is selected or what Datasets are loaded. Open the Control Panel by clicking on the **Control Panel** button on the Toolbar or by selecting the menu item **Windows>Control Panel**. The Control Panel groups controls in six tabbed folders. Select a particular control by clicking on the appropriate tab at the top of the window.



- **Variable** allows you to choose a variable in your model and select it as the Workbench Variable.
- **Time Axis** allows you to change or focus the period of time over which Analysis tools operate.
- **Scaling** enables you to change the scales of output graphs.
- **Datasets** allows you to manipulate the stored datasets (runs).
- **Graphs** brings up the Custom Graph Control.

3

A Hands-On Example

Modeling with Vensim

Modeling using the following steps provides maximum effectiveness in using Vensim:

- Construct (or open an existing) model.
- Examine the structure using the structural Analysis tools (Tree Diagrams, etc.).
- Simulate the model.
- Examine the behavior in the model using the dataset Analysis tools (Graphs and Tables, etc.).
- Perform simulation experiments to understand or refine the model.
- Present the model and its behavior to your audience using Analysis tool output or custom output from the Graph Control.

Constructing, examining, and modifying models should follow an iterative approach. Starting from simple models with few feedback loops and little detail allows the quick construction of a working simulation model. The working model can then be modified and improved as necessary to show the desired level of detail and complexity.

Vensim has a unique approach to displaying simulation output. During a simulation, dynamic behavior is stored for all variables in the model. The user then selects the variable that they want information about and clicks on the appropriate Analysis tool to display the information.

The Workforce Inventory Example

In this Chapter you will work through the mechanics of using Vensim with a workforce inventory model. This is a simple, but quite valuable model to study. It demonstrates how the interaction of inventory management policies and hiring practices can lead to instabilities in production. It also demonstrates the somewhat counterintuitive result that being more aggressive about hiring and laying people off can actually lead to a more stable workforce. The workforce inventory model is developed in Chapter 16.

Starting Vensim

Windows 3.1, 3.11 or NT 3.xx

- ▶ Double click on the **Vensim icon** appearing in the program group the icon installed in.

Windows 95, 98, NT 4.0 or 2000


- ▶ Click on the **Start** button then **Programs>Vensim>Vensim PLE Pluss32** (or Vensim PLE32 and so on).

Macintosh

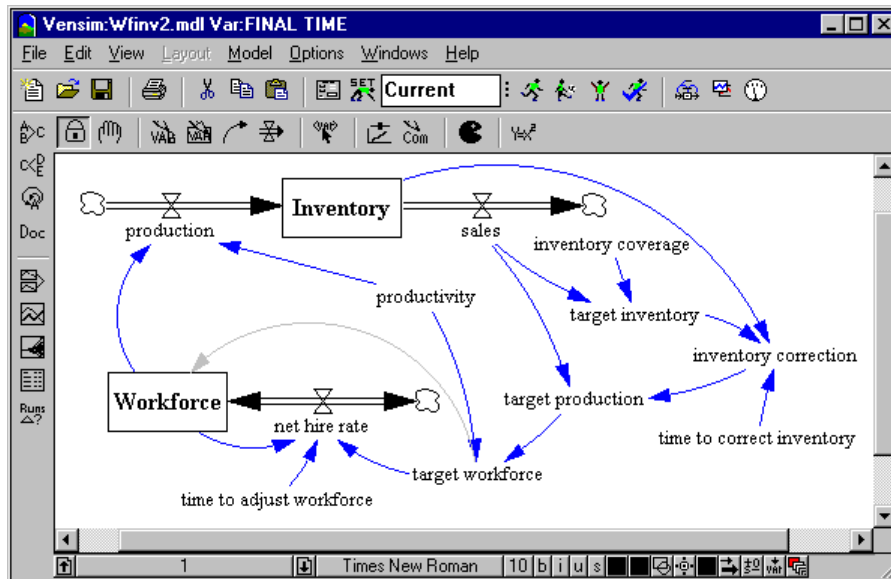
- ▶ Double click on the **Vensim icon**.

Vensim will open with a new (empty) model. We could start developing our model here, but instead we will open and simulate an existing model.

Opening the Model

- ▶ Select the menu item **File>Open Model...**, or click on the **Open Model** button  on the Toolbar.
- ▶ Open the model *wfinv.mdl* located in the directory *plemodel\chap03* (normally the full path is *c:\Program Files\Vensim\plemodel\chap03* but we will omit the earlier part).

Vensim will load the Workforce/Inventory model and the screen should appear as below.



This model describes the dynamic behavior of a manufacturing plant that carries inventory. The Title Bar displays the model that is loaded (*wfinv.mdl*) and the Workbench Variable (*Workforce*). We can see that the variable *Workforce* also appears in the sketch. The Workbench Variable is any variable in the model that we are currently interested in focusing on. We can change the focus any time we want by double clicking on another variable.

- The Sketch **Lock** tool should be selected by default. Place the mouse cursor over the box in the sketch that says *Inventory*, then double click on it.

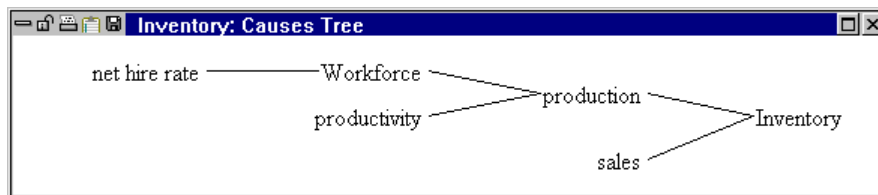
We see that the Workbench Variable (on the Title Bar) changes from *Workforce* to *Inventory*.

Examining Structure




The workforce/inventory model presented is relatively simple, although it may look baffling at first. In this visual representation, arrows imply cause and effect: the variable at the tail of the arrow causes the variable at the head of the arrow (to change). For example, *production* is caused by *Workforce* and also by *productivity*.

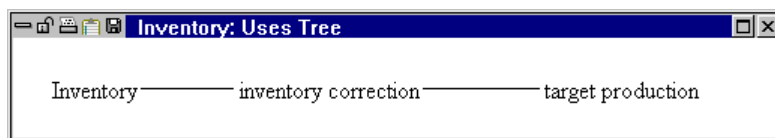
We can investigate the structure of this model with the structural Analysis tools. We will get answers only about structure, not about dynamic behavior of the model (that comes next when we simulate the model and use the dataset Analysis tools).

- Click on the top Analysis tool, the **Causes Tree Diagram**  and an Output window opens:




We see that the Workbench Variable, *Inventory*, is on the right and everything that causes it to change (up to 2 connections distant) is on the left.

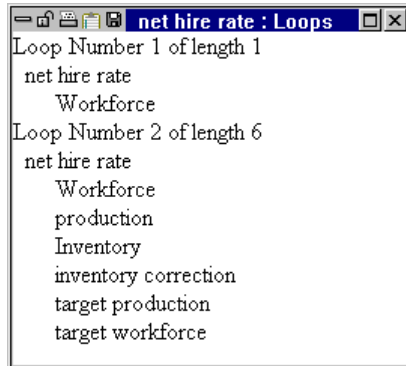
- Click the **Close** button  in the upper left corner, or the **Close** button  in the upper right corner, or press the Del key, to close the Tree Diagram.
- Click on the **Uses Tree Diagram** Analysis tool  and an Output window opens:




Now we can see the Workbench Variable on the left and where it is used in the model (what it causes to change, up to 2 connections distant) on the right. Note that these Tree Diagrams simply present information from the model in a different manner. We can observe all the causal connections by examining the sketch, but trees make parts of the model more presentable and easier to understand.

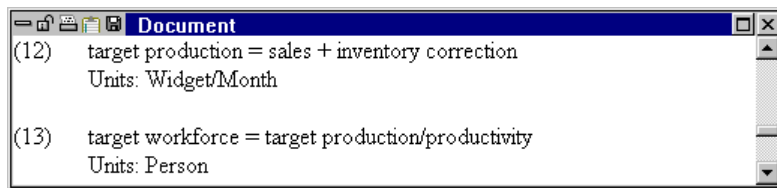
3: Vensim PLE User's Guide

- ▶ Click the **Close** button or press the Del key to close the Tree Diagram.
- ▶ Place the mouse cursor pointer on the variable *net hire rate* that appears in the sketch, then double click to select it as the Workbench Variable.
- ▶ Click on the **Loops** Analysis tool 



An Output window opens that displays all variables in all feedback loops (two) that pass through the Workbench Variable (*net hire rate*).

- ▶ Click on the **Document** Analysis tool 



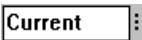
An Output window opens that displays the equations and units of measure for all the variables in the model. This tool does not use the Workbench variable.

- ▶ Select the menu item **Windows>Close All Output**.


This closes all the Output windows that have been created.

Simulating the Model

Now we would like to examine the dynamic behavior of the model. We want to look at the behavior of variables in the model, such as the amount of *Inventory* over time. To do this, first we need to simulate the model.


- ▶ Double click on the simulation **Runname** editing box on the Toolbar  to highlight the default name *Current* (or click once and drag over the name *Current*), then type in the name

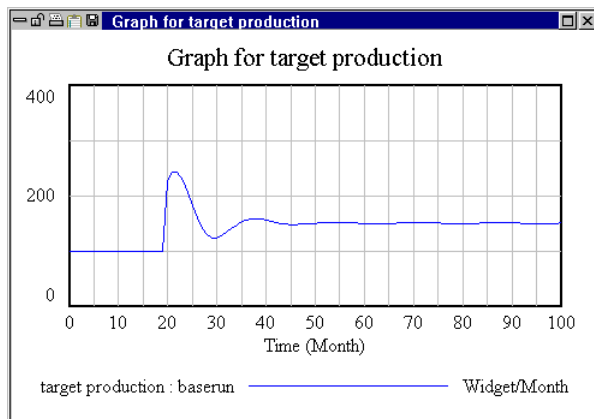
baserun. This is the name of the dataset that holds all the simulation output values for behavior of variables when we make a simulation run.

- ▶ Click on the **Simulate** button  on the Toolbar. Vensim will simulate the model and store the values for the dataset *baserun*.


Examining Behavior

Now we can look at the dynamic behavior of the workforce/inventory model. The dataset *baserun* has stored the behavior of every variable in the model.

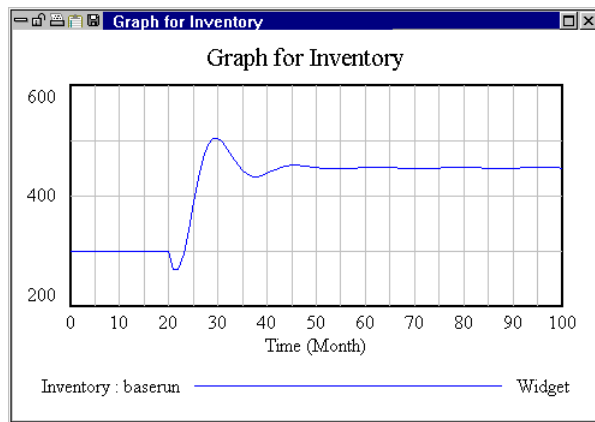
- ▶ Double click on *target production* to select it as Workbench Variable.
- ▶ Click on the **Graph** Analysis tool 



Note the oscillating behavior (repeated increasing and decreasing) for *target production*, which climbs from an initial value of 100 then oscillates and settles to 150.

- ▶ Close the graph by clicking on the **Close** button or pressing Del.
- ▶ Double click on the variable *Inventory* appearing in the sketch and then click on the **Graph** tool 

3: Vensim PLE User's Guide



We see a graph of *Inventory* with oscillating behavior similar to *target production*, although *Inventory* starts out by declining before increasing in value. Let's look at a table of the actual values for inventory.

- Click on the **Table** tool 

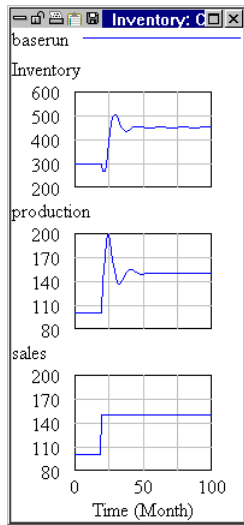
Time (Month)	0	1	2	3	4	5
"Inventory" Runs:	baserun					
Inventory	300	300	300	300	300	300

- Use the bottom scrollbar of the Output window to look through the values for *Inventory*.

Now remember how we looked at the causes of *Inventory* by using the **Causes Tree Diagram** Analysis tool? We can also look at graphs of behavior of the variables that cause *Inventory* to change.

Causal Tracing

- Click on the **Causes Strip** Analysis tool 



A strip graph is generated that shows the Workbench Variable (*Inventory*) at the top, and all the variables that directly cause *Inventory* to change below it (*production* and *sales*).

Notice something very interesting in this graph. *Inventory* has oscillating behavior which then is damped out and becomes stable. *Inventory* is being changed by both *production* and *sales* but *only production* is oscillating. *Sales* does not have the oscillating pattern of behavior contained in *Inventory* and *production*. Therefore we will look into *production* and not *sales*.

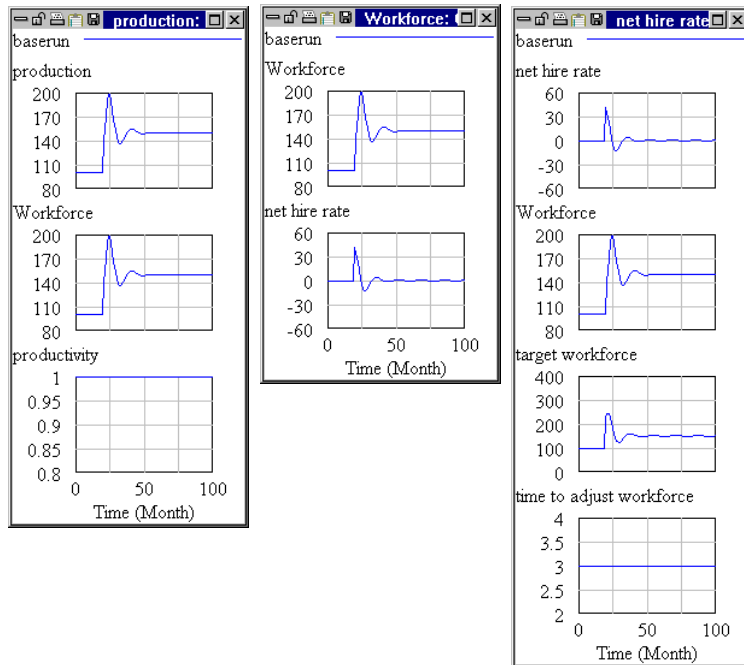
Causal Tracing is a quick and powerful tool that helps us determine *what* portions of a model are causing *which* types of behavior. The **Causes** and **Uses Tree Diagrams** and the **Table** tool can all be used for Causal Tracing. The most commonly used tool is the **Causes Strip** tool and we will use that to investigate the sources of oscillation in this model.

Now let us track the oscillation to find out which feedback loops in the model are causing the oscillating behavior.

3: Vensim PLE User's Guide

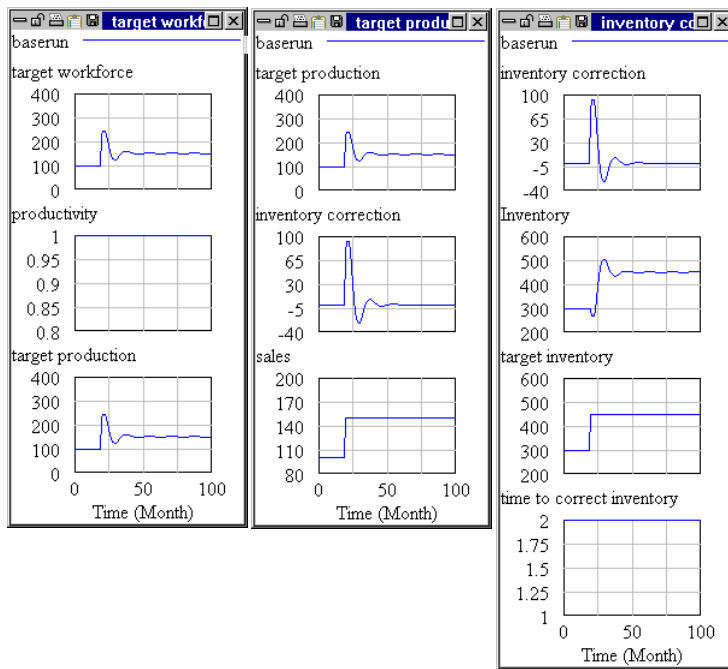
- ▶ Double click on *production* appearing in the **Causes Strip** to select it as the Workbench Variable, then click on the **Causes Strip** tool.
- ▶ Double click on *Workforce* in the **Causes Strip** that has just been displayed, then click on the **Causes Strip** tool.
- ▶ Double click on *net hire rate* then click on the **Causes Strip** tool.

The three strip graphs are displayed below. Note how the oscillation is traveling through all these variables.




- ▶ Double click on *target workforce* then click on the **Causes Strip** tool.
- ▶ Double click on *target production* then click on the **Causes Strip** tool.
- ▶ Double click on *inventory correction* then click on the **Causes Strip** tool.

3: A Hands-On Example



The last two graphs show similar behaviors. The **Causes Strip** for *target production* shows that the oscillation is coming from *inventory correction*, not from *sales*. In the *inventory correction* graph, we see that *Inventory* is causing the oscillation, not *target inventory*.

We know that the oscillations follow a path back to *Inventory* that do not go through the variable *sales*. Let's look at the sketch to get a feel for what is happening.

- Click on the **Build Windows** button  on the Toolbar.

This brings the Build window to the front and pushes the Output windows to the back.

With your eyes, trace the feedback loop that the oscillations have followed, from *Inventory* to *production* to *Workforce* to *net hire rate* to *target workforce* to *target production* to *inventory correction* and back to *Inventory*.

Look at the variable *target production*. Note how the oscillations travel through the feedback loop to *Inventory*, not through *sales*. The variable *sales* is a Constant with a STEP function. *sales* causes other variables to change, but nothing causes it to change. *sales* is not part of any feedback loop. The variable *sales* imparts the sudden change to the Level *Inventory* (through a step increase in *sales*). The system structure (the negative feedback loop) then tries to correct *Inventory* and sets up the oscillation at some particular frequency. This is very much like a rocking chair that will rock back and forth in response to a push in one direction.

- Select the menu item **Windows>Close All Output**.

Simulation Experiments

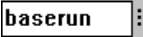


- Click on the **Set Up Simulation** button  on the Toolbar.

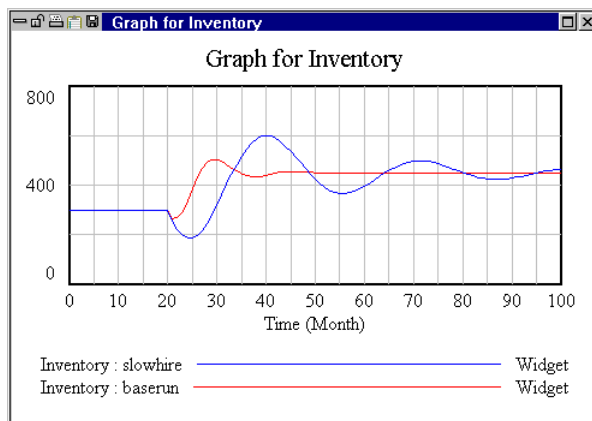
We see some of the variable names in the sketch appear with yellow text on a blue background. These are Constant variables that do not change during simulation; we can set them to a different value before we simulate and see the effect the changes have on model behavior.

- Click on the variable *time to adjust workforce* that appears yellow/blue on the sketch.

An editing box will open:

We will try an experiment where we slow the rate at which we hire new workers (and layoff current workers), to see if that removes the oscillation. Ideally we would like to see a smooth increase from our old inventory (and workforce) levels to the new levels.

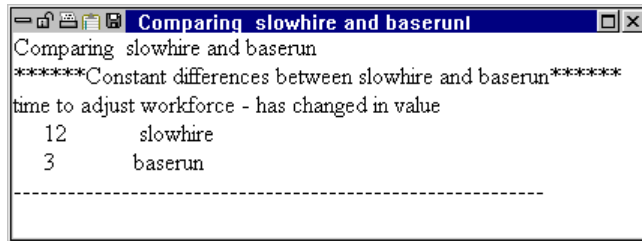
- Type the number 12 into the editing box to replace the number 3 (for a 12 month delay), then press the Enter key.
- Double click on the name **baserun** to select it in the simulation **Runname** editing box on the Toolbar  and type in the name *slowhire*.
- Click on the **Simulate** button  on the Toolbar, the model will simulate and store the values for the dataset *slowhire*.
- Double click on the variable *Inventory*, then click on the **Graph** Analysis tool 



Here we see the results of two experiments: *baserun* with the original value (3) for the variable *time to adjust workforce*, and *slowhire* with the modified value (12) for *time to adjust workforce*. The results show that slower hiring and firing practices actually increase the size of the oscillation, and make the oscillations last longer.

To see what the differences in the constants were for each run:


- Click on the **Runs Compare** Analysis tool 

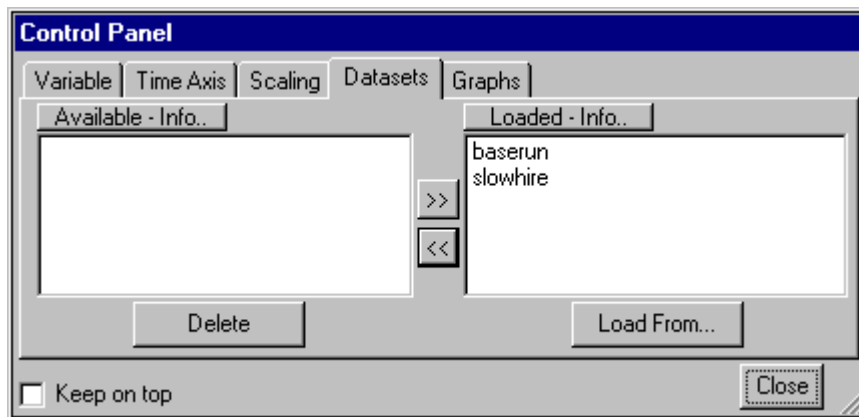


The **Runs Compare** tool lists all the Constant (and any Lookup) differences in the loaded simulation datasets. We have two datasets loaded (*baserun* and *slowhire*) and the only difference is the value for the variable *time to adjust workforce* (3 and 12).

Making a Custom Graph

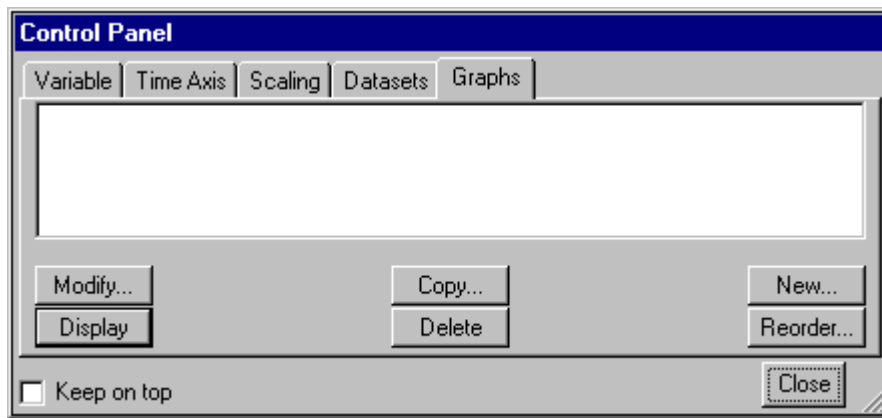
Sometimes you will want to see all of the important variables together in one graph. Graphs generated using the Analysis tools display behavior for only the Workbench Variable. Using Custom Graphs, you can display the desired variables, dataset runs, style and formatting in one graph. **Custom Graphs** are created from the Graph Control located in the Control Panel.

- Click on the **Control Panel** button  on the Toolbar to select the Control Panel. Click the tab for **Datasets**.

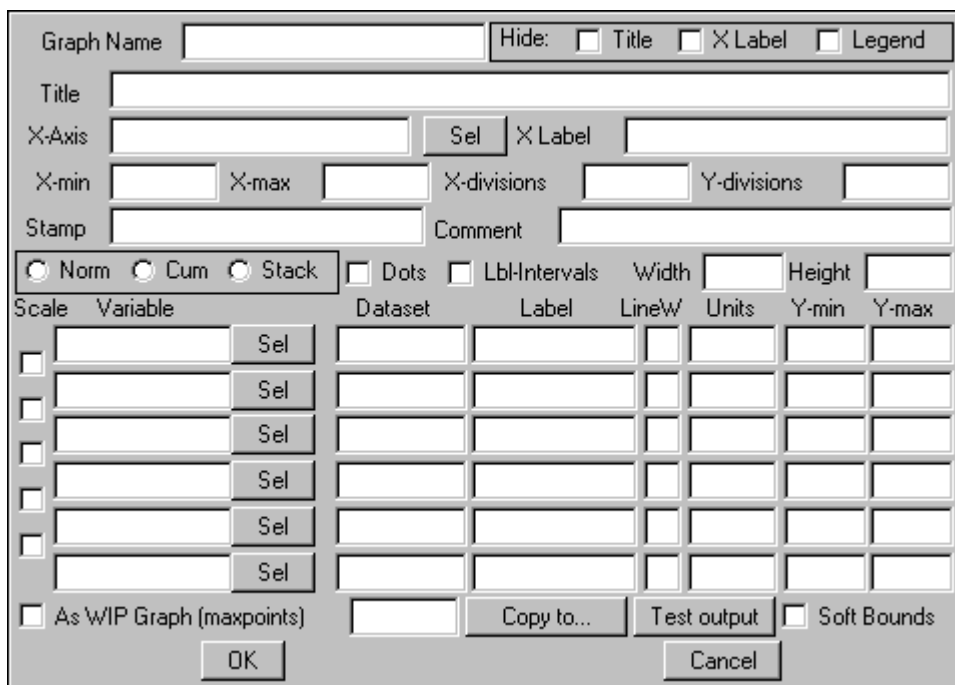


- Unload baserun by double clicking on the run name baserun in the **Loaded** runs box.
- Click on the tab **Graphs** in the Control Panel.

3: Vensim PLE User's Guide

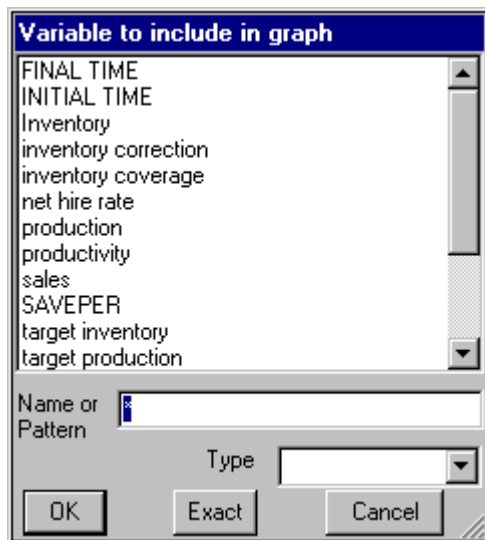


- Click the button **New....** The Custom Graph Editor opens with the cursor positioned at the graph **Title** editing box.

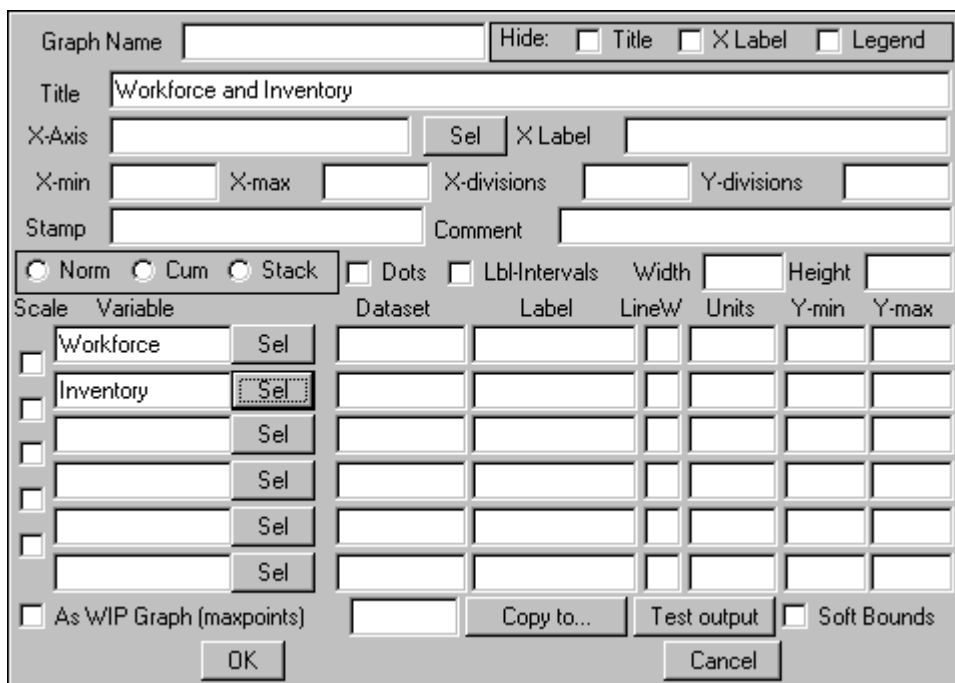


- Type the name Workforce and Inventory into the **Title** editing box.
- Using the mouse, move to the **Variable** boxes on the left side of the graph editor and click on the top button labeled **Sel**. A variable selection dialog box appears.

3: A Hands-On Example

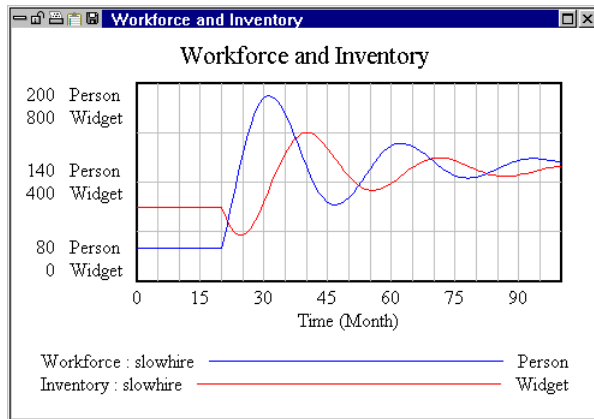


- Move the scrollbar down the list and double click on *Workforce*.
- Using the mouse, click on the second button down labeled **Sel**. A variable selection dialog box appears, move the scrollbar down the list and double click on *Inventory* (or single click and click **OK** to close the variable selection dialog).



3: Vensim PLE User's Guide

- ▶ Click the **OK** button to close the Custom Graph Editor.
- ▶ Click the button **Display** in the Graph Control to show the Custom Graph.



Continue On Your Own

If you wish, you can continue to trace structure or behavior of this model, or make additional runs (experiments) with different Constant values.

4 Causal Loop Diagrams

Vensim Models

This chapter describes causal loop diagramming. These diagrams do not show accumulations (levels or stocks) in a system, so if you want to add this detail, refer to Chapter 5 "Stock and Flow Diagrams." Causal loop diagrams are called that because each link has a causal interpretation. An arrow going from A to B indicates that A causes B.

Vensim diagrams are constructed by entering information graphically into a sketch view. Simulation models can be built from diagrams by entering algebraic relationships for each variable using the Equation Editor (see Chapter 6 "Building a Simulation Model").

In Vensim Professional and DSS, models and equations can also be entered directly using the Text Editor. Graphical diagrams, based on the text model, can be constructed from text equations.

Drawing Sketches

When a Sketch Tool is selected, that tool remains active until you select another tool. A single click (press and release) with the mouse button applies the tool to the sketch.

The **Lock** tool provides the standard mouse cursor. The Lock tool can be used for selecting sketch objects (they highlight black) and for changing options. Sketch objects cannot be moved with the Lock tool. Tip — you can select the Lock tool by pressing the Esc key, or the keyboard number 1.

The **Move/Size** tool is used for moving sketch objects around, including resizing variables and boxes, and reshaping arrows. The other sketch tools also allow you to move objects.

Variable sketch tools (**Variable**, **Box Variable**, etc.) and the default setting for **Rates** bring up editing boxes (for naming the Variable or Rate) when applied to the sketch. The **Sketch Comment** tool brings up a dialog box.

Arrows are started with a single click (press and release) of the mouse button, then finished with another single click. Arrows (curved) can take one intermediate point on a sketch with an extra mouse click. (*Do not* try to draw arrows by clicking and holding the mouse button down while dragging the mouse.)

Objects in a sketch can have their appearance changed by clicking on them with the right mouse button, which brings up an options dialog box.

Mouse Tips

- If a mouse button click is called for without mention of left or right, use the left button (Macintosh, use the only button).
- If a mouse right button click is called for, with the Macintosh, hold down the Control key or the Apple key and click (Ctrl + Click or \mathbb{A} + Click).

Constructing a Causal Loop Diagram

This section describes the construction of a causal loop diagram of a project. A central concept is the amount of work to do that is left in the project. Much of our diagram will center on this concept. First we will construct a diagram with one view that describes some essential elements of getting a project done. A view is a single sketch of your model, like a single page of a book. Your model can contain multiple views. Later, we will add another view that incorporates more of our knowledge of the system.

Project Model (*project.mdl*)

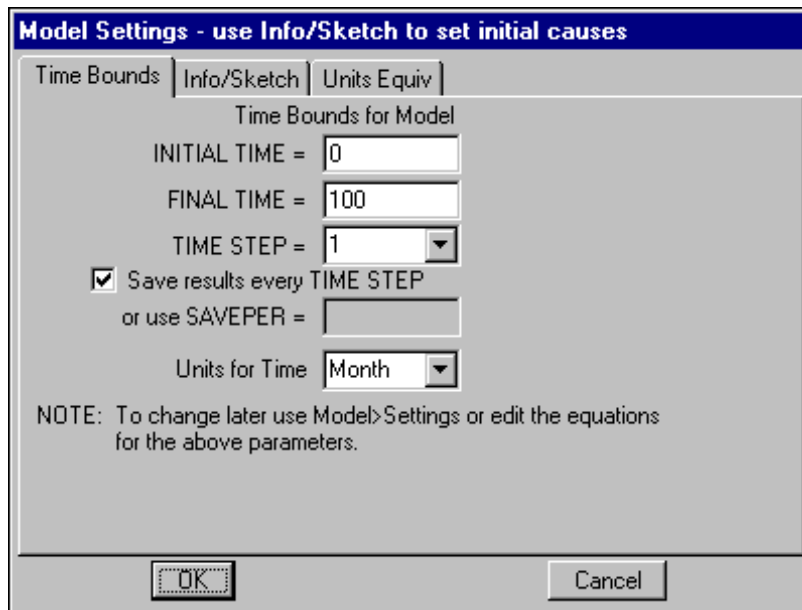
This model describes the competing feedback loops in a project model. The causal loops show the relationship between the amount of *Work To Do* and *overtime hours required*, and how increased overtime affects both the amount of *work done* and also the amount of *fatigue* experienced by the workforce. The first cut at this model assumes a constant size of workforce.

- Start Vensim

Vensim will open with the last model you worked on active.

- Select the menu item **File>New Model** or click the **New Model** button on the Toolbar.

The Model Settings dialog box opens:



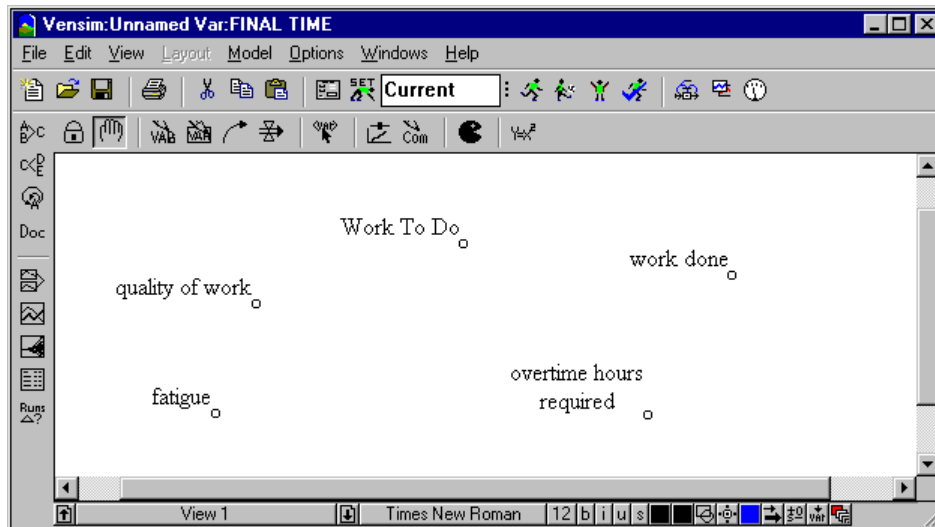
- Click **OK** to accept the default values

A causal loop diagram does not use the Time Bounds, but a simulation model needs Time Bounds and therefore all Vensim models (including diagrams) require Time Bounds by default.

Click the **Save** button on the Toolbar. Select the directory *plemodel\chap04* then type in the name *project* and click the **Save** button in the dialog box.

Adding Variables

- Click with the mouse button on the **Variable** tool (or press the keyboard number 3 above the letter keys — not the numerical keypad).
- Click in the middle top of the sketch and type *Work To Do* in the editing box, then press Enter.
- Click again on the sketch and continue filling out the diagram with the variables shown below.



Moving Sketch Objects

- ▶ Select the **Move/Size** tool by clicking on it (you can also press the keyboard number 2). Move the mouse directly over a variable. Press down and hold the mouse button then drag the mouse. A box will move to show you the new position for the variable. Release the mouse button and the variable will move to the new position.

You can also move and reposition objects using other sketch tools.

- ▶ Select the **Variable** tool again (click on it, or press the keyboard number 3). Move the cursor directly over a variable. Press down and hold the mouse button then drag the variable to a new position.
- ▶ Return the variables to the positions shown in the diagram above.

Now that we have laid out some important variables, let us show their causal influences.

Adding Arrows

- ▶ Select the **Arrow** tool by clicking on it (or press the keyboard number 5). Click once on *Work To Do*. Be sure to let the mouse button up without moving the mouse! Move the cursor to *overtime hours required* and click again. A straight arrow will join the two variables.
- ▶ Click once on *overtime hours required* then move the cursor to *work done* and click again. A straight arrow will join the two variables.

Handles

Handles are the little circles that appear in the middle of arrows in Vensim sketches, and at the corner of boxes and clear boxes, in the middle of rates and elsewhere. These handles allow you to resize or move things around. These handles appear when first entering variables, when creating arrows, and

4: Causal Loop Diagrams

any time the **Move/Size** tool is selected. Handles can be manipulated by the **Move/Size** tool or any other sketch tool except **Lock**.

- Select the **Move/Size** tool to turn handles on.

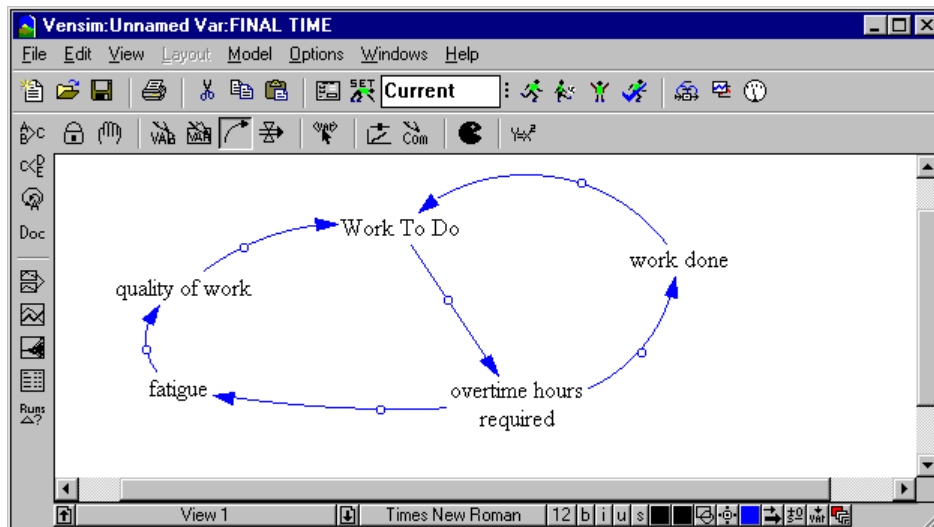
Curved Arrows

One Way:

- Position the pointer on top of the handle in the middle of the straight arrow from *overtime hours required* to *work done*. Press and hold the mouse button down, then drag the mouse (and arrow) a down to make a curved arrow.

Another Way:

- Select the **Arrow** tool. Click once on *work done*, then move the cursor to a blank portion of sketch just above and right of *Work To Do* and click once, then move the cursor onto *Work To Do* and click once again. A curved arrow will join the two variables. You can move this arrow by dragging the handle (with the **Arrow** tool or the **Move/Size** tool).
- Continue joining variables with curved arrows, according to the diagram below, by either making straight arrows and moving the handle to curve them, or by making a single intermediate click on the sketch.



Editing Variables

- To edit a variable name, click on it with the **Variable** tool to open the editing box, then type in a new name.

Deleting Variables

If you want to delete a variable from the model you can use either **Edit>Cut** (Ctrl + X) or press the Del key on the keyboard (both of which open a prompt dialog) or use the **Delete** tool (which deletes from the model with no prompt). If you want to remove a shadow variable from the view (but still leave it in the model), select menu **Edit>Cut** or press Del and click the button **Remove from this view but do not change model structure** in the prompt dialog.

- ▶ Select the **Variable** tool and click on the sketch, then type in the name *temporary* and press Enter.
- ▶ Select the **Delete** tool and click on the variable *temporary*.

Undo and Redo

If you make a mistake while creating a model, you can use the menu item **Edit>Undo** and **Edit>Redo** commands to step backward or forward. Ctrl+Z is the same as Edit>Undo and Ctrl+Y is the same as Edit>Redo. The undo/redo history has multiple levels for most editing changes.

Saving Your Model

- ▶ Click the **Save** button or select the menu item **File>Save** or press Ctrl + S. Save the model in the directory *plemodel\chap04* with a name such as *project*.

The model will be saved in text format, with the file extension *.mdl*. Vensim PLE Plus also supports saving to a binary format - see Chapter 11.

Modifying diagrams

Sketch objects have options which you can change. These options allow you to customize your sketch. Two different methods are used to change sketch options:

1. Click with the right mouse button on a sketch object (for the Macintosh, Ctrl + Click)
2. Select the sketch object (variable, arrow, etc.) then use the Status Bar to change the options or attributes of the selected object.

Selecting Sketch Objects

Several methods allow you to select single or multiple sketch objects.

- Click on a single object with the **Move/Size** tool.
- Select multiple objects by holding the mouse button down and then dragging the **Move/Size** (or **Lock**) tool over a region of sketch.
- Select multiple objects by holding down the Shift key and clicking on each object with the **Move/Size** tool.
- Select the whole sketch with **Edit>Select all** (or Ctrl + A).
- Deselect objects which are selected by holding down the Shift key and clicking on each object with the **Move/Size** tool.
- Deselect all objects by clicking on a blank region of sketch (outside of the selection rectangle).

Sketch Layout

Vensim includes menu commands to help you lay out your sketch in a tidy manner. These commands allow you to resize sketch objects to default values, line up objects by position with a "last-selected" object, size objects to the last-selected, and more.

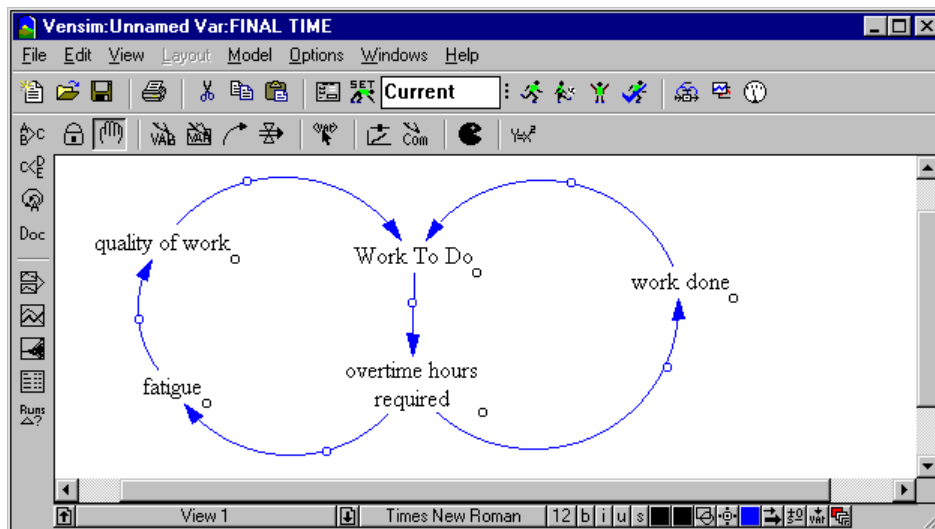
We shall tidy up the diagram, centering most variables on *Work To Do*.

- ▶ Select the **Move/Size** tool by clicking on it or pressing keyboard number 2.
- ▶ Click once on *overtime hours required* then hold the Shift key down and click once on *Work To Do*. Select menu **Layout>Center on LastSel**.

overtime hours required will move to line up on the center of *Work To Do*.

- ▶ Click once on *quality of work* then hold the Shift key down and click once on *Work To Do*. Select menu **Layout>Vertical on LastSel**.
- ▶ Click once on *fatigue* then hold the Shift key down and click once on *quality of work*. Select menu **Layout>Center on LastSel**.
- ▶ Click once on *fatigue* then hold the Shift key down and click once on *overtime hours required*. Select menu **Layout>Vertical on LastSel**.
- ▶ Drag *work done* to right of and halfway between *Work To Do* and *overtime hours required*.
- ▶ Move the arrows to make neat curves resembling a circle (see below).

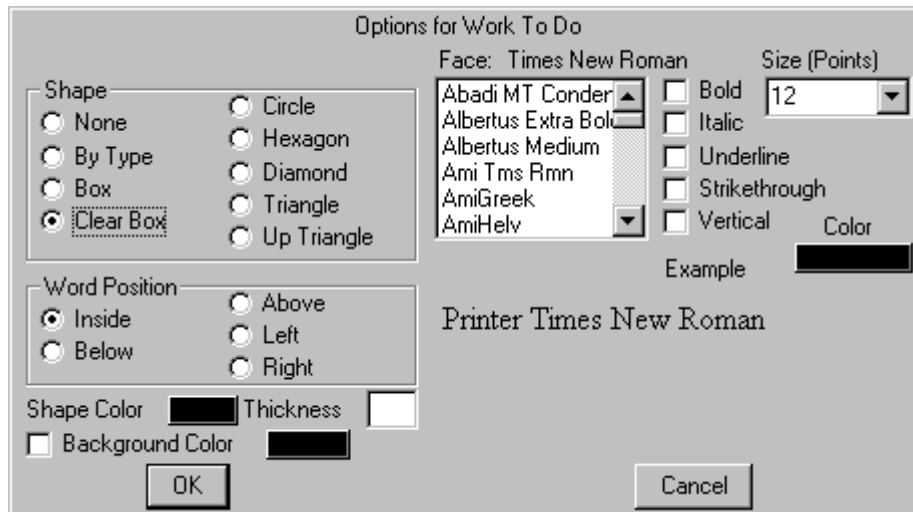
Your sketch should look something like:



Sketch Options

Variables

- Select the **Lock** tool. Use the right mouse button single click on the variable *Work To Do*. For the Macintosh, hold the Control key down and click with the mouse button (Ctrl + Click).



An options dialog box will open.

- Change the font (e.g., to Arial), the size, color, or anything else, then click **OK**.

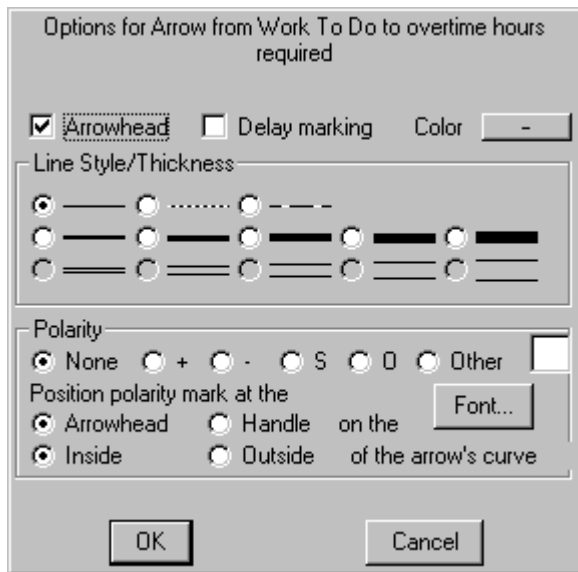
Note that in the options dialog the **Word Position** option only applies if the variable has a **Shape** selected (anything but **None**).

- Select menu **Edit>Select all** or press Ctrl + A. Click on the font size button on the Status Bar at the bottom of the window (probably reads **12**) and choose a bigger size, say 14. Click outside of the highlighted box.

Arrows

- Click with the right mouse button on the arrowhead of the arrow from *Work To Do* to *overtime hours required*, an options dialog box will open:

4: Causal Loop Diagrams

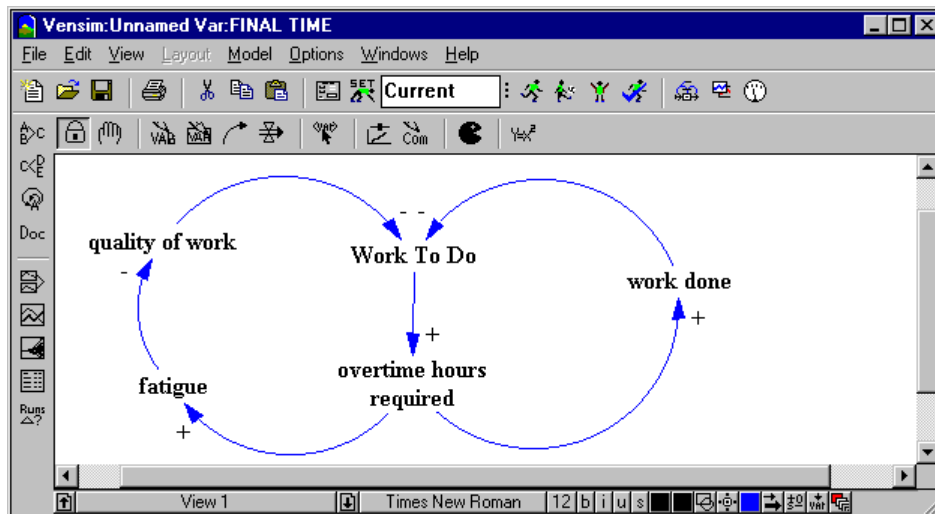


As *Work To Do* increases, *overtime hours required* also increase, a positive causality.

- Select + (under **Polarity**) and **Outside** (of the arrow's curve) then click **OK**.

The polarity (+) is, by default, attached to the head of the arrow inside the curve.

- Continue changing the polarity of arrows according to the figure below, selecting **Outside** of the arrow's curve for each arrow:




Now you will highlight the positive feedback loop with thicker and colored arrows.

- Select the **Move/Size** tool if it is not already selected.

4: Vensim PLE User's Guide

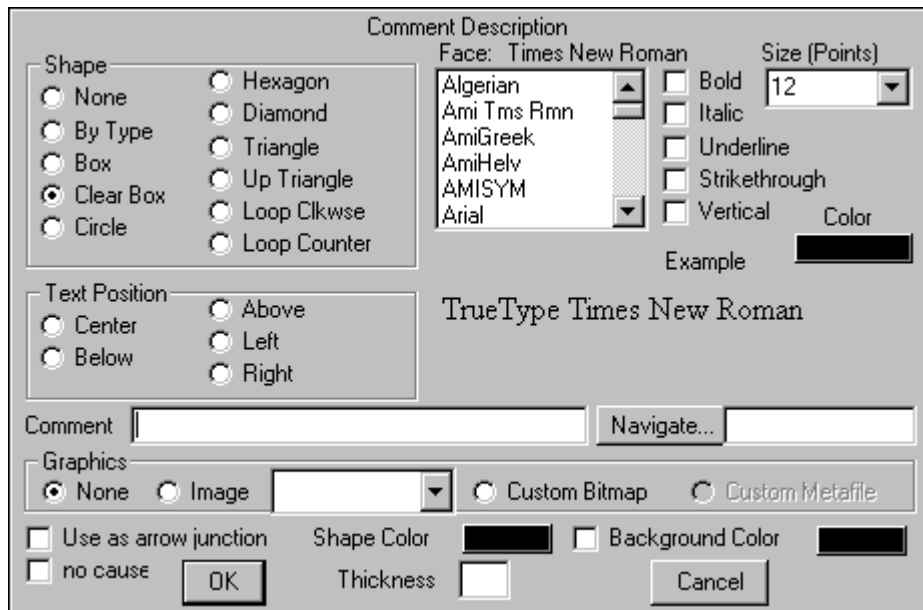
- ▶ Click once on the arrowhead of the arrow from *Work To Do* to *overtime hours required*. Now hold down the Shift key and click on the other arrowheads of arrows from:
 - *overtime hours required* to *fatigue*
 - *fatigue* to *quality of work*
 - *quality of work* to *Work To Do*

This will highlight all the handles and show a dotted box around the perimeter of the selected arrows.

- ▶ Release the Shift key.
- ▶ On the Status Bar, look for the button with two arrows of different widths . Click on it and choose the fifth line from the top. All arrows highlighted will increase in width.
- ▶ Click on the **color** button just to the left of arrow width (probably colored blue) and choose a different color (e.g., red). Now click somewhere on the diagram outside of the dotted box to unselect the arrows.

Adding Comments and Graphics



- ▶ If you need to make some room at the top of your diagram, select the **Move/Size** tool, then choose menu **Edit>Select all** (or Ctrl + A), then using the cursor, drag the whole diagram lower on the view to make room for the title.
- ▶ Select the **Sketch Comment** tool. Click at the top of your sketch to add a title; the Comment dialog box will open.



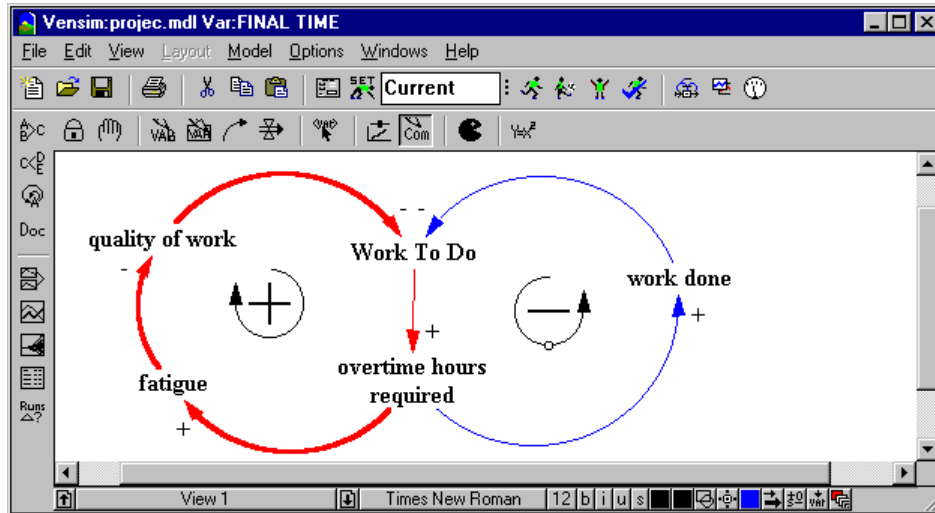
The image shows a 'Comment' dialog box with various settings for text and graphics. The 'Shape' section has radio buttons for 'None', 'By Type', 'Box', 'Clear Box', 'Circle', 'Hexagon', 'Diamond', 'Triangle', 'Up Triangle', 'Loop Clkwise', and 'Loop Counter'. The 'Text Position' section has radio buttons for 'Center', 'Below', 'Above', 'Left', and 'Right'. The 'Comment' field is empty, and there is a 'Navigate...' button next to it. The 'Graphics' section has radio buttons for 'None', 'Image', 'Custom Bitmap', and 'Custom Metafile'. There are also checkboxes for 'Use as arrow junction', 'no cause', 'Shape Color', 'Background Color', and 'Thickness'. The 'Face' dropdown is set to 'Times New Roman', and the 'Size (Points)' is set to '12'. The 'Example' text shows 'TrueType Times New Roman'.

- ▶ Type in a name for your sketch (e.g., Work To Do Project Model). Choose a font, size, color, shape and word position for your comment, then click **OK**.

4: Causal Loop Diagrams

- Still with the **Sketch Comment** tool, click on the sketch in the center of the left hand loop. Click on the dropdown arrow in the **Graphics** field **Image** box and choose the positive sign (+) or the snowball image , then from the **Shape** field choose **Loop Clkwse** (clockwise), and click **OK**.
If you need to, reposition the loop image and resize the loop by dragging its handle. Note that you can also include a bitmap or metafile from the clipboard.
- Click on the sketch in the center of the right hand loop. Click on the dropdown arrow in the **Image** box and choose the negative sign (—) or the balance image , then from the **Shape** field choose **Loop Counter** (counter clockwise), and click **OK**. If you need to, reposition the loop image and resize the loop by dragging its handle.

Your diagram should now look similar to the figure below:



Refining the Model

We see that *overtime hours required* increases the *work done*, which then decreases *Work To Do* (a negative feedback loop). However, *overtime hours required* also drives the positive feedback loop that increases *Work To Do*. To make this model more realistic, let us assume that we can change the size of the workforce by hiring or firing workers. Hiring workers will reduce the need for overtime and therefore reduce the fatigue/Work To Do spiral.

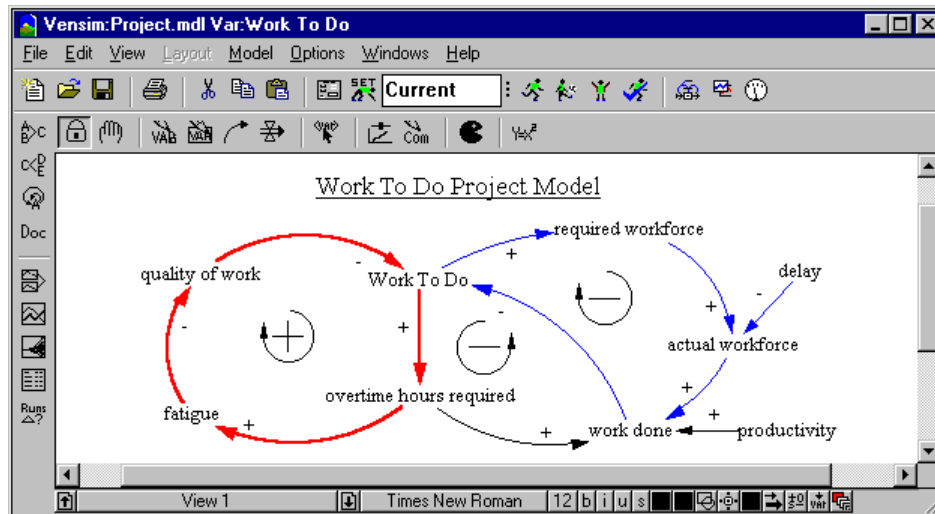
An Additional Feedback Loop

- Select the **Move/Size** tool and move the variable *work done* lower on the sketch. Reshape the arrows to and from *work done*, and move the negative loop symbol, as shown in the diagram below.
- Select the **Variable** tool. Click on the sketch and type in the new variables *required workforce*, *actual workforce*, *delay*, and *productivity* (pressing Enter after each variable) as shown in the diagram below.

4: Vensim PLE User's Guide

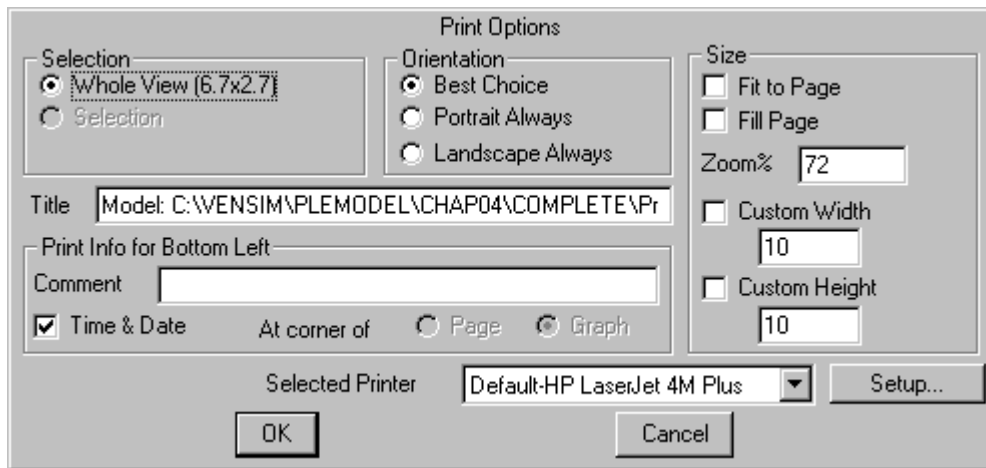
- ▶ Select the **Arrow** tool and connect the variables as shown in the diagram below.
- ▶ Optionally, add polarities to the arrows and change arrow thickness.
- ▶ Select the **Move/Size** tool, click once on the negative feedback symbol to highlight it, then choose **Edit>Copy** (or Ctrl + C). Select **Edit>Paste** (or Ctrl + V), then click on **OK** or press the Enter key in the dialog to choose **Replicate**. The new loop image gets pasted on top of the first image. Drag the copied feedback loop from the old location and place it in the center of the new feedback loop. Click on it with the right mouse button (Macintosh: Ctrl + click) then under **Shape** choose **Loop Clkwise** then click **OK**.

Your diagram will finally look something like this:



Printing and Exporting the Sketch

The sketch can be printed by clicking the **Print** button or by selecting the menu item **File>Print** while in the Build window.



The Print Options dialog gives a number of options, the more important ones are:

- **Selection** — print whole view, or print selected (portion of view)
- **Orientation** — portrait or landscape orientation
- **Size** — fit to page will fit your view onto a single page.
- **Title** — this will appear on the top of the printed page.

The sketch can be exported to the clipboard for use in other applications by using **Edit>Select all**, or selecting a group of variables with the **Lock** tool, then selecting **Edit>Copy** (Ctrl + C). This exports the sketch information to the clipboard as a metafile, which can then be pasted into other applications.

Structural Analysis of Diagrams

Analysis Tools

Vensim Analysis tools all into two broad classes: tools for structural analysis, and tools for dataset analysis. Structural tools allow the user to investigate the model structure; dataset tools allow the user to investigate simulation datasets to determine the behavior of variables. In this section, we will analyze the structure of our model. Structural Analysis tools include the **Tree Diagram** tool (**Causes Tree** and **Uses Tree**), the **Loops** tool, and the **Document** tool.

Analysis tools almost always work by generating information about the Workbench Variable. You select the Workbench Variable by one of two methods. The easiest method is to double click on the variable, wherever it appears. The variable is usually somewhere in one of the sketches, unless the model is text-based. You can also double click on a variable in an Output window, such as a **Tree Diagram** or a **Strip Graph**. The second way to select a variable into the workbench is to click the **Control Panel** button to open the Control Panel, select the tab **Variable** to open the Variable Selection Control, then choose the variable from the list. The Workbench Variable always appears on the title bar of the model.

4: Vensim PLE User's Guide

NOTE If you activate an Analysis tool that requires a simulation dataset, and you have no simulation dataset loaded, you will see the message "No runs are loaded. Please load runs". This tells you that you need to run a simulation. Chapter 5 will describe how to build a simulation model.

EITHER

- Use the model *project.mdl* that you just built,

OR

- Open the model in the *plemodel\chap04\complete* directory.
- Select the **Lock** tool. Now move to the variable *Work To Do* and double click on it to select it as the Workbench Variable. The title bar should look like:

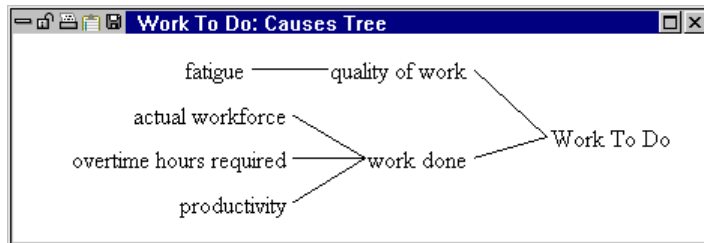


Causal Tracing[®] with Trees

Causal Tracing is a powerful tool for moving through a model tracing what causes something to change. Causal Tracing Analysis tools can be configured to show the causes of a variable or the uses of a variable (the opposite direction to causes).

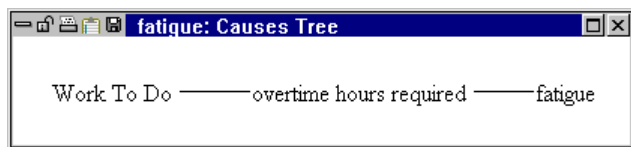
Causes Tree Diagram

- Click on the **Causes Tree** tool. We see the causes of *Work To Do*:



We can trace through the diagram looking at what causes any particular variable.

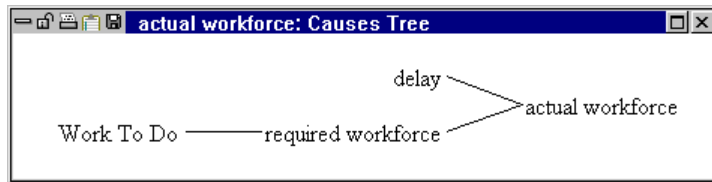
- Double click on *fatigue* appearing in the tree diagram window then click on the **Causes Tree** tool again:



We can see that *fatigue* is caused by *overtime hours required* and by *Work To Do*. Now we have traced all the way around one feedback loop, starting and finishing at *Work To Do*. Let us look at what causes *actual workforce*.

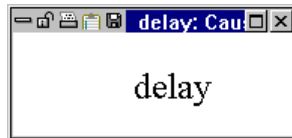
4: Causal Loop Diagrams

- Double click on *actual workforce* appearing in the first tree diagram or on the sketch then click on the **Causes Tree** tool:



We have traced another causal loop, from *Work To Do* through *actual workforce* and back to *Work To Do*.

- Double click on *delay* in the tree diagram then click on the **Causes Tree** tool:

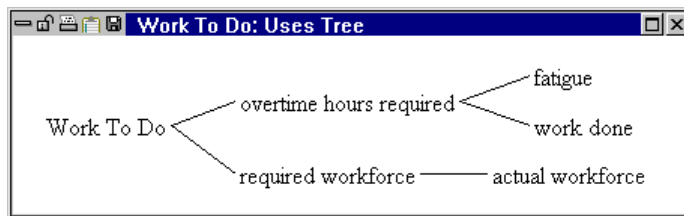


There are no causes of *delay*; it is a Constant in this model.

Uses Tree Diagram

Now let us look at a **Uses Tree** diagram.

- Double click on *Work To Do* to select it as the Workbench Variable (check the title bar).
- Click on the **Uses Tree** tool.



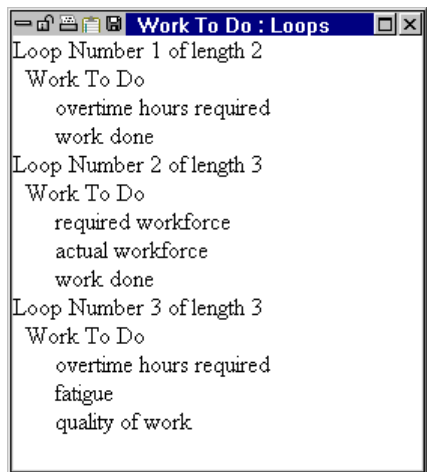
Uses are the opposite of causes, you see where in the model *Work To Do* is used.

Loops Tool

Now let us look at a tool that finds feedback loops for you: the Loops tool.

- Make sure *Work To Do* is still the Workbench Variable (look at the title bar).
- Click on the **Loops** tool. *Work To Do* is involved in three feedback loops:

4: Vensim PLE User's Guide



Document Tool

This tool provides a text-based description of all equations in your model.

- Click on the **Document** tool.

A description of your entire model is generated. Note that the time bounds for the model are included, even though this is not a simulation model.

5 Stock and Flow Diagrams

Stock and flow (or Level and Rate) diagrams are ways of representing the structure of a system with more information than a simple causal loop diagram. Stocks (Levels) are fundamental to generating behavior in a system; flows (Rates) cause stocks to change. Stock and flow diagrams are a common step toward building a simulation model because they help define types of variables that are important in causing behavior. We will construct a diagram describing the relationships among awareness of a product and the quantity of customers and potential customers.

Building a diagram (*customer.mdl*)

- ▶ Start Vensim.
- ▶ Click on the **New Model** button (or select menu **File>New Model...**) and click **OK** in the Model Settings dialog box to accept the default values.
- ▶ Click the **Save** button on the Toolbar. Select the directory *plemodel\chap05* then save as *customer* or some other name of your choice.

Entering Levels

Levels are also known as stocks, accumulations, or state variables. Levels change their value continuously over time (without instantaneous steps). Rates, also known as flows, change the value of levels. In turn, levels in a system determine the values of rates. Intermediate concepts or calculations are known as auxiliaries and, like rates, can change instantaneously.

When constructing a Level and Rate diagram, consider what variables accumulate over a period of time. Another way to think about this: if Time slowed down to zero for your system, what variables would still retain a value? For example, in the system where you pour water into a glass, the water contained in the glass is the Level. If you froze time, the pouring (a Rate) would stop, but you would still see a quantity of water in the glass (a Level). Once you know what levels you need, enter them first and then connect the rates and auxiliaries. Model building tends to be iterative. Don't try to get everything right first time; you can always change things later on.

- ▶ Select the **Box Variable** tool and click once on the diagram. Type in *Potential Customers* and press Enter.
- ▶ With the **Box Variable** tool still selected, click on the diagram approximately 3 inches (7 cm) to the right of *Potential Customers*, type in *Customers*, and press Enter.

Creating Rates

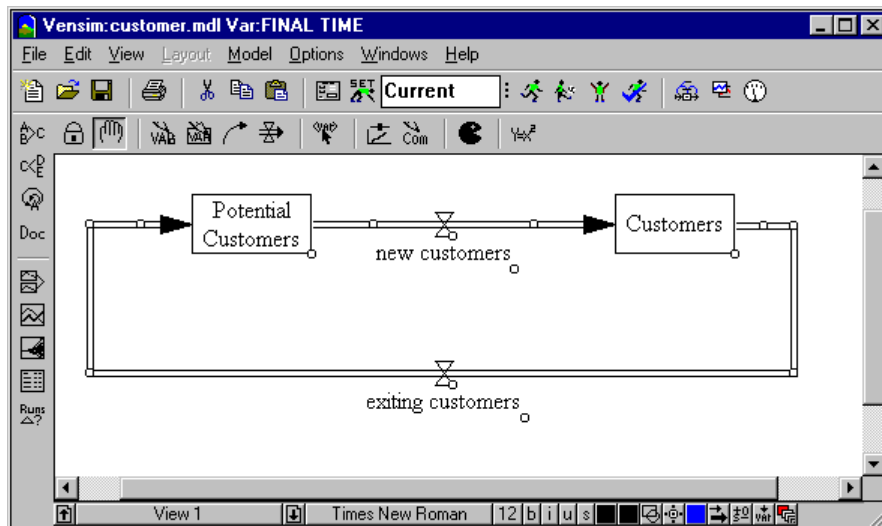
- Select the **Rate** tool. Click once (click and release the mouse button) on top of *Potential Customers*, then move the cursor on top of *Customers* and click once again. Type the name *new customers* and press Enter.

The Rate has a single arrowhead, indicating the direction that material can flow (the Rate can only increase the Level). Note that this is only a diagram. In a simulation model, the equation governs the direction that material can flow. Also note that the default setting for the **Rate** tool can be changed to enter two arrowheads (pointing both directions).

Bending Rate Pipes

- Click one on *Customers*, press and hold the Shift key down then move the cursor about half an inch (1 cm) to the right of *Customers* and click once. Continue holding down the Shift key for all further clicks. Move directly down about an inch and a half (3.5 cm) (below and right of *Customers*) then click once. Move directly left to 1 inch below and left of *Potential Customers* and click once. Move directly up just left of *Potential Customers* and click once. Move on to *Potential Customers* and click once. Release the shift key. Type the Rate name *exiting customers* and press Enter.

Your diagram should look like the one below:



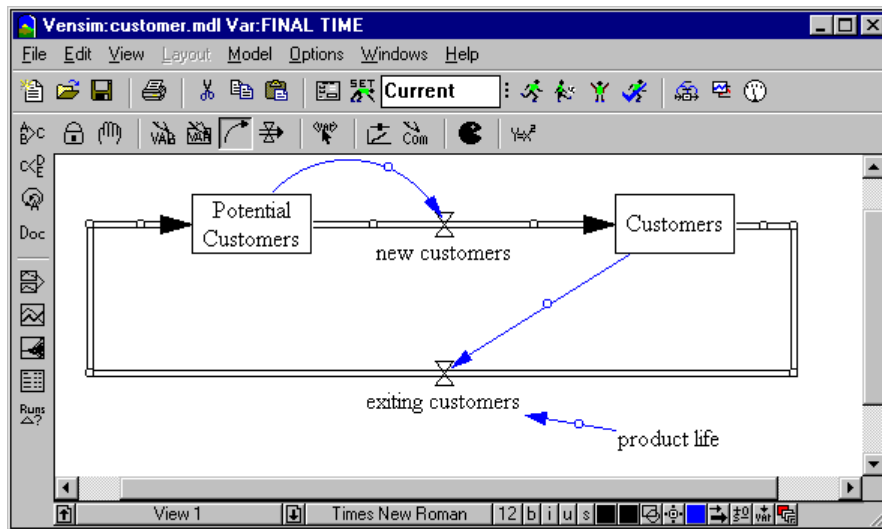
Adding Auxiliaries and Arrows

- ▶ Select the **Variable** tool. Click once just left and below *new customers* then type the name *time to become customer* and press Enter.
- ▶ Click once just right and below *exiting customers* and type the name *product life* and press Enter.
- ▶ Select the **Arrow** tool. Click once on *Potential Customers*, then once on a blank part of the sketch above and between *Potential Customers* and *new customers*, then once on the valve attached to *new customers*.

A curved arrow will join the Level and the Rate valve. Vensim allows you to connect arrows to either the Rate name or the Rate valve. The Rate name and valve are structurally the same.

- ▶ Click once on *time to become customer* then once on the Rate name *new customers*.
- ▶ Click once on *Customers* then once on the valve attached to the Rate *exiting customers*.
- ▶ Click once on *product life* then once on the Rate name *exiting customers*.

Your diagram should now look like the one below:



More Structure

- ▶ Select the **Box Variable** tool and click once on the diagram approximately 2 inches (5 cm) above the Level *Potential Customers*. Type in the name *Awareness* and press Enter.
- ▶ Select the **Rate** tool. Click once on an empty part of the diagram about 2 inches (5 cm) left of the Level *Awareness*. Move the cursor over to *Awareness* and click once on this Level. Type in the variable name *gaining awareness* in the editing box then press Enter.

5: Vensim PLE User's Guide

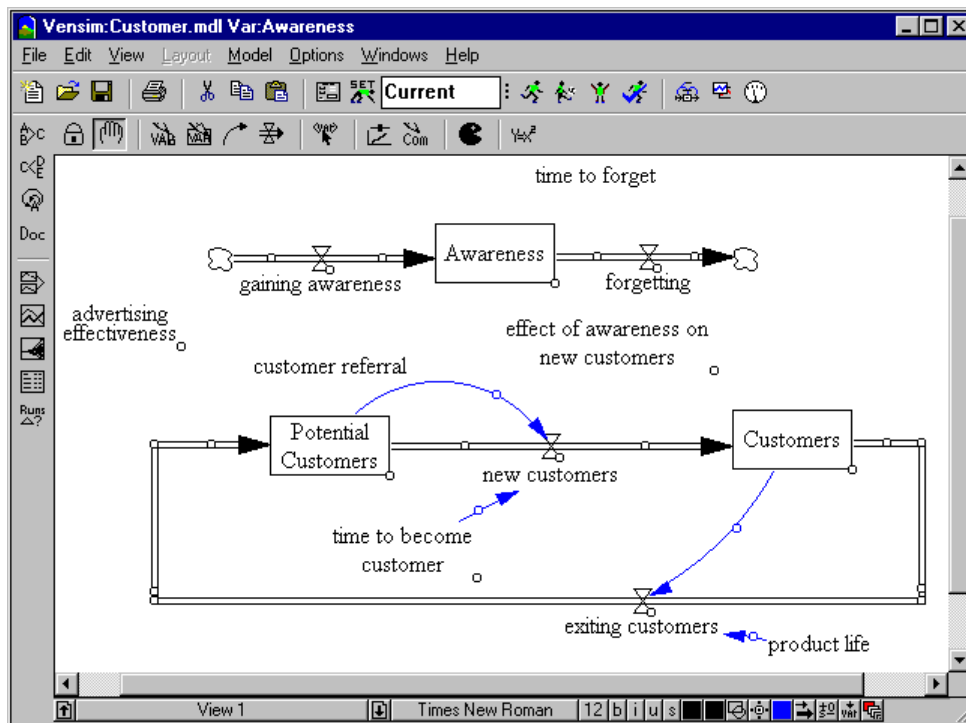
A Rate named *gaining awareness* will be constructed between the Level and a cloud. The cloud defines the limits of the model; we do not care where the material comes from, or what happens if the material goes into a cloud.

- ▶ Click once on the Level *Awareness*. Move the cursor to an empty part of the diagram about 2 inches to the right of *Awareness* then click once. Type in the Rate name *forgetting* in the editing box, and then press Enter.
- ▶ Select the **Variable** tool. Click on the sketch below and left of the Rate *gaining awareness* and type the name *advertising effectiveness* but **do not** press Enter (leave the editing box open).
- ▶ Click again on the sketch just below *advertising effectiveness* and type *customer referral*.
- ▶ Click again on the sketch just above the Rate *forgetting* and type *time to forget*.

Note how you can enter multiple variables by clicking on the sketch *without* pressing the Enter key.

- ▶ Click again on the sketch between *Awareness* and *new customers* and type *effect of awareness on new customers* and press Enter.
- ▶ If you are running out of space, you can move the whole structure around the page. Choose menu **Edit>Select All** (or press Ctrl + A) then move the structure around the page (you cannot do this if the **Lock** tool is selected, use the current Sketch tool or the **Move/Size** tool). Click outside the structure to unselect it.

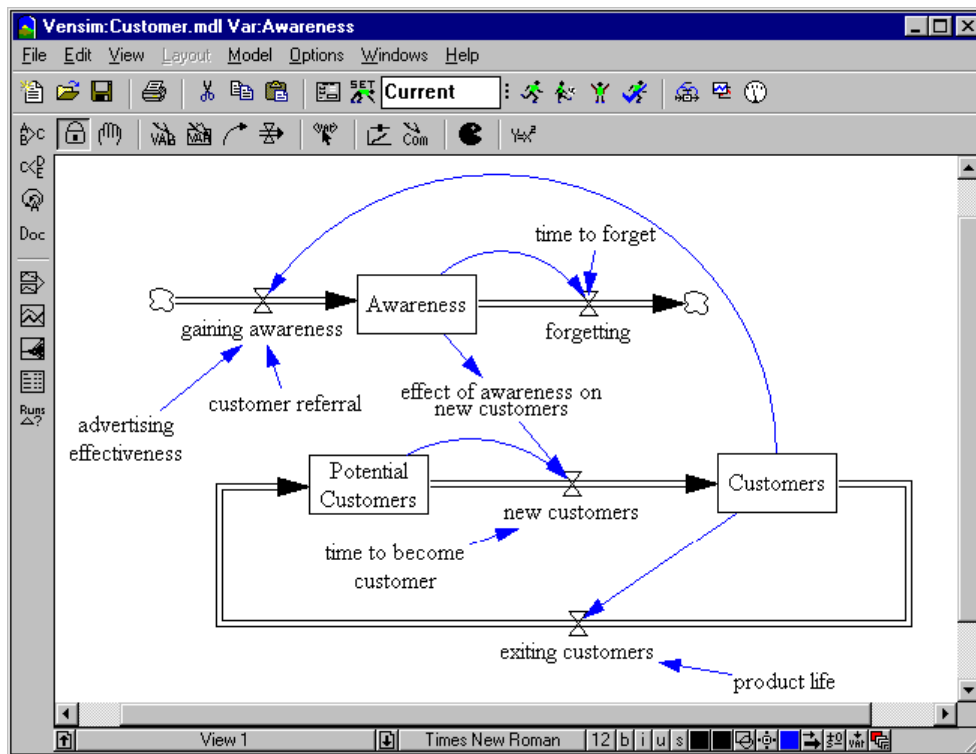
Your sketch should now look like this:



- ▶ Select the **Arrow** tool. Click once on *advertising effectiveness* then once on *gaining awareness*.
- ▶ Click once on *customer referral* then once on *gaining awareness*.
- ▶ Click once on *Awareness* then once on a blank part of the sketch above and between *Awareness* and *forgetting*, then once on the valve attached to *forgetting*.
- ▶ Click once on *time to forget* and then once on the valve for *forgetting*.
- ▶ Click once on *Awareness* then once on *effect of awareness on new customers*.
- ▶ Click once on *effect of awareness on new customers* then once on the valve for *new customers*.
- ▶ Click once on *Customers* then once on a blank part of the sketch a little above the cloud for the Rate *forgetting*, then once on *gaining awareness*.
- ▶ Move variables around and drag the arrow handles with the **Move/size** tool if you need to make the sketch neater.

Your sketch should now be complete and look like this:

5: Vensim PLE User's Guide



- Click the **Save** button on the Toolbar to save your work.

Customizing Diagrams

Diagrams can be customized in many different ways. The standard practice for Vensim diagrams is to show levels (stocks) as a box, with the name inside the box. Rates are shown with the Rate valve explicitly named, although sometimes a Rate will be unnamed (show only a valve). Auxiliaries, constants, lookups, data (exogenous) variables, etc. are shown as a simple name. This section describes some options for customizing diagrams.

Sketch Options

- Select the **Move/Size** tool. Click on the handle of the Level *Awareness* (little circle at the bottom right corner of the box) and drag it until the box is a little larger.
- Click once on the Level *Potential Customers* to select it, then hold the Shift key down and click once on *Customers* and once on *Awareness*. Select menu **Layout>Size to LastSel**.
- Select menu item **Edit>Select all** (or Ctrl + A). Click on the button for setting color of arrows on the Status Bar (find this button by holding the cursor over each button until the name "set color on

selected arrows" appears). Choose a color (e.g., red) then click on the sketch outside of the selection box.

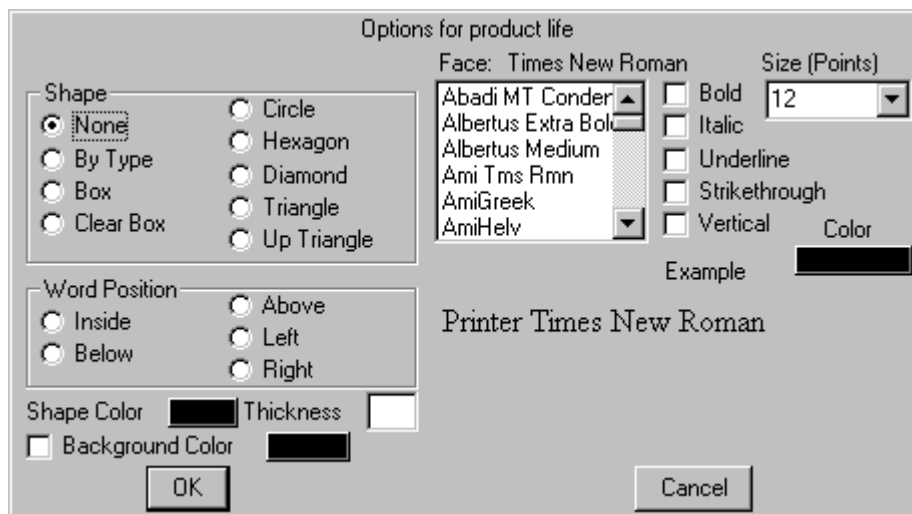
Note that both arrows and rates are changed. If you want to change arrows and rates separately, you need to select each type individually while holding the Shift key (e.g., select all the rates then choose a color, then repeat for arrows). Or you can select all, then unselect the ones you don't want to change.

- ▶ Click on *Awareness* then hold the shift key down and click on *Potential Customers* and *Customers*. On the Status Bar, click on **b** to select bold (toggles to **B**), and if you wish, change the color of the variables and the box color (also located on the Status Bar).
- ▶ Press the Esc key or keyboard number 1 to return to the **Lock** tool (or click on the **Lock** tool).

Variable Shapes

We can change the shape of an individual variable by clicking on it with the right mouse button, then selecting a new shape in the Options dialog box.

- ▶ Click with the right mouse button on the variable *product life*.



- ▶ In the Options dialog box, click on the option button for **Circle** in the **Shape** field, click on the button for **Below** in the **Word Position** field then click **OK**.

An alternative way to do this is to click on the variable with the **Lock** or **Move/size** tool to highlight it, then select a shape from the **Surround shape** button on the Status Bar.

5: Vensim PLE User's Guide

Resizing the View

- ▶ Select **View>Zoom>200%**.

The sketch will zoom to 200 percent.

- ▶ Select **View>Zoom>100%**.

The sketch will zoom back to the original size. Zooming simply focuses in or out of the sketch, making everything larger or smaller.

6 Building a Simulation Model

A Population Model

This chapter features a simulation model of a rabbit population. The modeling process starts with sketching a model, then writing equations and specifying numerical quantities. Next, the model is simulated with simulation output automatically saved as a dataset. Finally, the simulation data can be examined with Analysis tools to discover the dynamic behavior of variables in the model.

Normal model construction follows a pattern of create, examine, and recreate, iterating until your model meets your requirements. Debugging (making a model simulate properly) and model analysis (investigating output behavior) both play a part in refining the model. Reality Check is another technology to aid in the construction and refinement of models and is described in Chapter 18.

The behavior of a simulation model in Vensim is solely determined by the *equations* that govern the relationships between different variables. We will list those equations in full for the simulation models developed in this Manual. The structural diagram of a model is a picture of the relationships between variables. Vensim enforces consistency of the diagram and model equations, but information can be omitted or hidden in diagrams. When you are building a simulation model, make sure the equations match what is in this manual. If there is a discrepancy in the appearance of the diagrams, check to be sure that **Show Initial Causes** is checked on the **Info/Sketch** tab of the Model Settings dialog (open this with the **Model>Settings** command).

Vensim Conventions

Naming

Model diagrams should be clearly presented to facilitate building, analysis, and presentation. Most of the models in this manual follow certain naming conventions that we recommend, though you may choose otherwise if you wish.

Levels have initial letters capitalized; e.g., *Population*

Rates, auxiliaries, constants, lookups, data variables, and other variable types are all lower case; e.g., *average lifetime*

Sketch

Levels or stocks are entered with the **Box Variable** tool. When using the **Box Variable** tool, the variable is designated as a Level. When you open the Equation Editor you will see that variables added with the **Box Variable** tool have type Level. You can change type in the Equation Editor, or make variables without boxes have type Level in the Equation Editor but this can cause confusion and is not recommended.

6: Vensim PLE User's Guide

Rates are usually entered with the **Rate** tool. By default, rates are added with a name. You can press the Esc key to close the editing box without adding a name. By default, Vensim also draws a Rate with one arrowhead, which indicates the positive direction of flow. You can add an arrowhead to the other end of the Rate by right-clicking on the handle (with the **Move/Size** tool selected) and checking the arrowhead box to indicate a two-way flow.

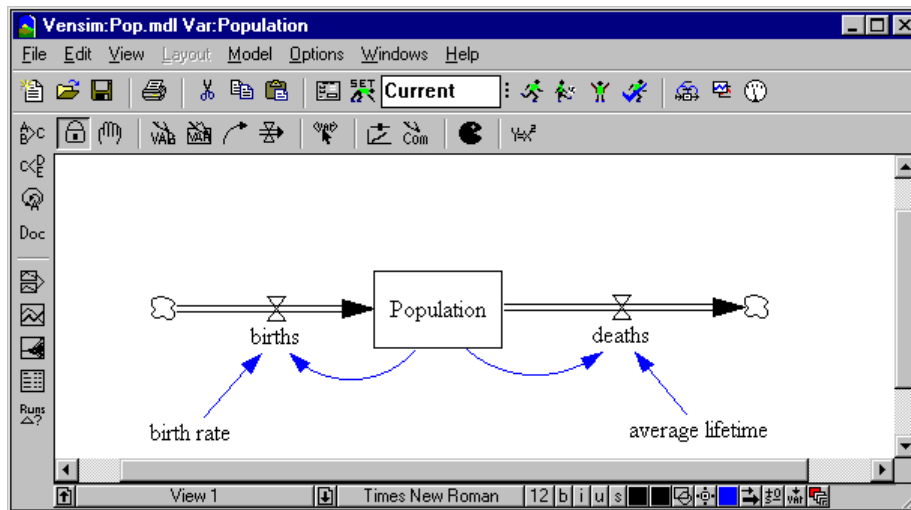
IMPORTANT NOTE The presence or absence of an arrowhead on a Rate has no effect on the equation for that Rate in a simulation model. The equation for a Rate could allow it to decrease a Level, even though a single arrowhead indicates on the sketch that the Rate increases that Level. It is up to the modeler to write an equation that makes the Rate behave properly.

Constants, Auxiliaries (and Lookups, Data, and other variables) are usually entered with the **Variable** tool as words in a clear box. Some diagramming conventions give Auxiliaries and Constants a Circle shape (usually with the name appearing below). You can change the shape around or above a variable by Right-Clicking (or Control-Clicking) on the variable name and choosing a different shape.

Sketching the Rabbit Model (*pop.mdl*)

- ▶ Start Vensim.
- ▶ Click the **New Model** button, or select the menu item **File>New Model...**
- ▶ In the Model Settings dialog (Time Bounds tab) type 30 for FINAL TIME, type (or select from the drop down box) 0.125 for TIME STEP. Click on the dropdown box for **Units for Time**, and select **Year**. Click on **OK** (or press Enter).
- ▶ Select the **Box Variable** tool and click somewhere in the middle of the sketch. Type the name *Population*, and press the Enter key.
- ▶ Select the **Rate** tool. Click once (single click and release of the mouse button) about 2 inches (5 cm) to the left of the Level *Population*, then move the cursor on top of *Population* and click once again. Type the name *births*, and press Enter.
- ▶ Click once on the Level *Population* then move the cursor about 2 inches (5 cm) right and click again. Type the name *deaths*, and press Enter.
- ▶ Select the **Variable** tool. Click on the sketch below *births*, type *birth rate* and press Enter. Click on the sketch below *deaths*, type *average lifetime* and press Enter.
- ▶ Select the **Arrow** tool, click once on *birth rate* then once on *births*. Click once on *average lifetime*, then once on *deaths*.
- ▶ Click once on *Population*, then once on the sketch a little below and left of *Population*, then once on *births*.
- ▶ Click once on *Population*, then once on the sketch a little below and right of *Population*, then once on *deaths*.
- ▶ Click the **Save** button and save your model in the directory *plemodel\chap06*. Name your model (we call it *pop.mdl*).

The structure of the Population model is now complete, as shown in the figure below. A positive feedback loop from *Population* to *births* increases *Population*, and a negative feedback loop from *deaths* decreases *Population*.



Writing Equations

The model is now structurally complete. However, in order to simulate, it needs a set of equations that describe each relationship. These equations are simple algebraic expressions, defining one variable in terms of others that are causally connected. For example:

$$births = Population * birth\ rate$$

Looking at the sketch view, *birth rate* has no causes; it is a Constant in the model. This Constant has a numerical value which we will fill in later.

We will fill in units of measurement for each equation we enter. Units allow us to check for dimensional consistency among all the equations. Dimensional consistency is important as a formal check of correct model structure. We will use the menu item **Model>Units Check** (Ctrl + U) to check the whole model after we have added all the equations.

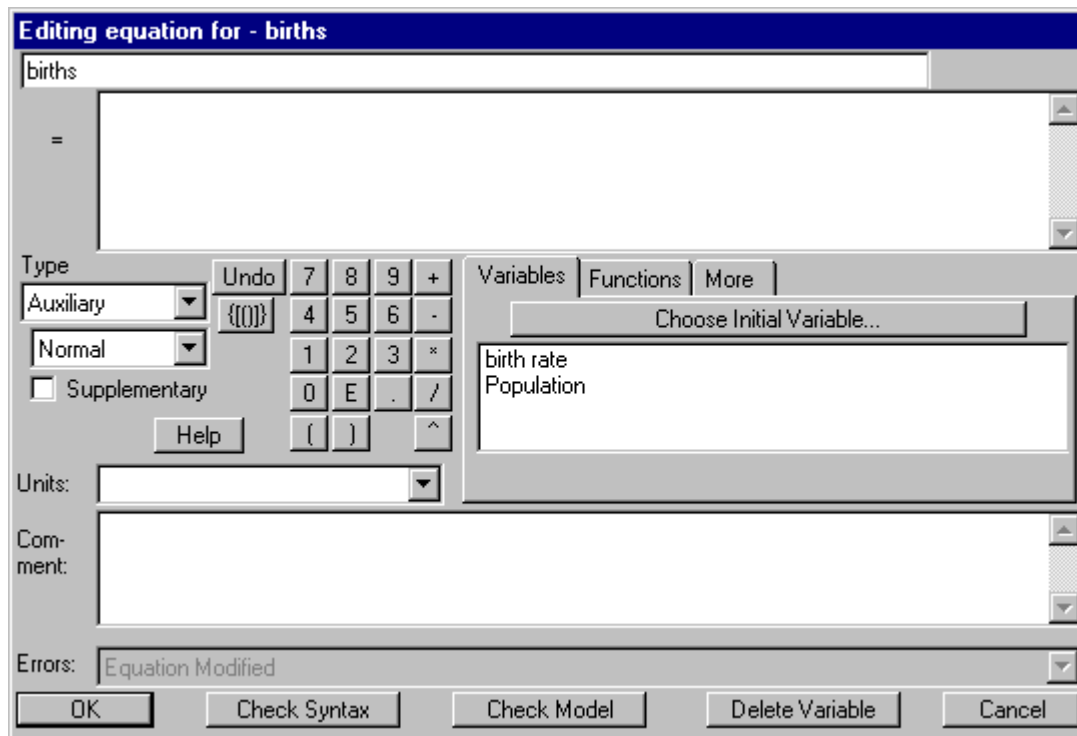
- Click on the **Equations** tool.

All the variables in the model will turn black. The highlights are used as a visible checklist of completeness. The highlights indicate which variables still require equations or have incomplete equations. As you complete the equations for each of the variables, the highlights will disappear. The menu item **Model>Check** (Ctrl + T) or the **Check Model** button in the Equation Editor also checks and displays what remains to be done.

- Click on the variable *births*.

Variable Type: Auxiliary

The Equation Editor opens. The top of the editor has the name of the variable we clicked on: *births*. The dropdown list box on the left shows the type of variable: **Auxiliary**. Vensim considers rates and auxiliaries to be the same type of variable. Click on the dropdown arrow to see the other types. Make sure that Auxiliary is still selected when you leave the list. Put the cursor in the equation editing box (next to the = sign).



- Complete the equation for births as below (in the editing box)

EITHER

- By typing *Population * birth rate*

OR

- Click on the variable *Population* in the **Variables** list (in the middle of the Equation Editor), then type the * symbol (or click on it in the Equation Editor keypad), then click on *birth rate* in the **Variables** list.

Spaces and new lines can be added to the equation for increased clarity, but are not necessary.

Now we will add the units of measurement for *births*.

- Type in the units of measurement *rabbit/Year* in the **Units:** box. This indicates that we measure the rate of *births* in rabbit per Year. Click on **OK** or press Enter.

If the model structure and equation agree and there are no syntax errors in the equation, the dialog box will disappear. If there are problems with the structure or equations, you will receive an error message indicating what is wrong.

Variable Type: Level

- Click on *Population*.

The Equation Editor opens and is slightly different from what we saw with the variable *births*.

The dropdown list box on the left shows the type of variable: **Level**. Left of the equation editing box is the INTEG function that defines a Level (integrating the variable over time). An equation is already present in the equation editing box. Because we connected rates with the names *births* and *deaths* to the Level, Vensim automatically enters the rates to the Level equation. Rates constructed by clicking first outside, and then on the Level are considered positive (inwards) flows; rates constructed by clicking on the Level, and then outside the Level are considered negative (draining) flows. If the rates are drawn in a different direction the sign can be changed in the Equation Editor. The equation for this Level is correct, *births* add to *Population*, *deaths* subtract from *Population*, so we need not change them.

The Equation Editor for a Level has an extra editing box to set the starting or initial value; the cursor is placed there.

- In the **Initial Value** editing box, type in 1000.

6: Vensim PLE User's Guide

This value is the initial number of rabbits at the start of the simulation (time zero).

- ▶ Type in the units of *rabbit* in the units box. Click on **OK** or press Enter.
- ▶ Click on *birth rate*. Type in the numbers 0.125 in the editing box.
- ▶ Type in the units *fraction/Year* (if you prefer, enter instead *1/Year*), this means that the fractional birth rate is measured in fraction (of rabbits) per year. Another way of saying this is (rabbits born / rabbit population) / Year (*rabbit/rabbit*) / Year = *fraction/Year* (rabbit cancels out). Click **OK** or press Enter.
- ▶ Complete the remaining two equations as they are shown in the Equations listing below.

All model variables should appear clear in the sketch after all the equations are entered. Units that have been previously entered can be selected again by clicking on the arrow in the units box and selecting from the dropdown list.

pop.mdl Equations

```
average lifetime = 8
Units: Year

birth rate = 0.125
Units: fraction/Year

births = Population * birth rate
Units: rabbit/Year

deaths = Population / average lifetime
Units: rabbit/Year

Population = INTEG(
    births - deaths,
    1000)
Units: rabbit
```

Checking for Model Syntax and Units Errors

Before we simulate the model, we should check it for errors in equations and units.

- ▶ Select **Model>Check Model** from the menu (or press Ctrl + T); you should get an information box saying "Model is OK."

If the model has errors, the Equation Editor will open with the variable containing the error. Check that the equation uses all the inputs and looks the same as in the listing above. Check that the structure of your model is the same as in the diagram above.

- ▶ Select **Model>Units Check** from the menu (or press Ctrl + U); you should get an information box saying "Units are AOK."

If a units error is generated, read the Output window to see which variables are failing the check. Open the **Equation Editor** on each variable and check the units against the list above. Units that do not check out often indicate poor or incorrect equation formulation.

NOTE The Units Check feature can also be accessed from the Analysis tools. You need to modify the toolset (menu **Tools>Analysis Toolset>Modify...**) and add the **Units** tool, or open the Analysis toolset *default2.vts* which contains the **Units** tool.

Units Equivalents (Synonyms)

When entering units, you may want to enter the plural form as well as the singular. For example, rabbits as well as rabbit. This will fail a Units check because Vensim does not see the plural form as the same word. The easiest way to accommodate this is to set rabbit and rabbits as equivalent units, or synonyms.

- ▶ Select menu **Model>Settings...** and click the **Units Equiv** tab. In the editing box, type `rabbit , rabbits` then click the button **Add Editing**. Then click **OK** to close the dialog.

Simulating the Model

- ▶ Double click on the **Runname** editing box on the Toolbar and type *equilib* for the first run name.
- ▶ Click on the **Simulate** button (or just press Enter when the cursor is in the **Runname** box).

The model will simulate, showing a work-in-progress window until completion. Note that on a fast computer the simulation may be so fast that the work-in-progress window may not be noticeable.

Model Analysis

This model has been designed to show equilibrium conditions in the rabbit population. The constants *birth rate* and *average lifetime* are both set to generate a rate of flow of 12.5 % of *Population*, therefore each feedback loop is balanced numerically, allowing no change in the value of *Population*.

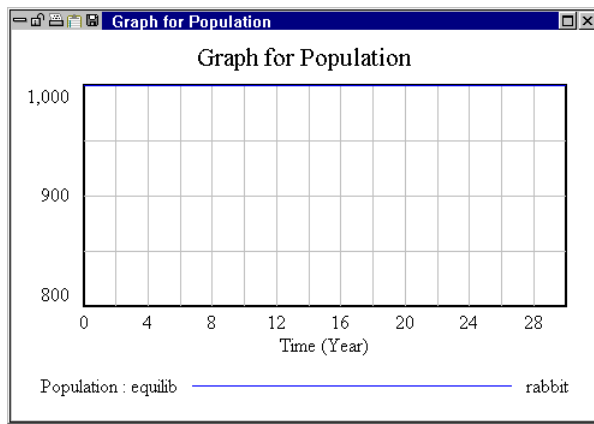
Graph and Table tools

- ▶ Double click on the Level *Population* in the sketch.

This selects it as the Workbench Variable; another way to do this is to choose *Population* from the Variable Selection control. Check the title bar at the top of the Vensim window to see that *Population* is selected.

- ▶ Click on the **Graph** tool. A graph of *Population* is generated:

6: Vensim PLE User's Guide



Population appears as a flat line at the top of the graph at 1000 rabbits. To check that no change is occurring:

- Click on the **Table** tool.

An Output window shows that *Population* is unchanging. Scroll the window to confirm that late in the simulation, *Population* is still 1000.

The figure shows a window titled "Table" with a table of simulation data. The table has columns for Time (Year) and Population. The data shows that the population remains constant at 1,000 rabbits throughout the simulation, even at the end of the run.

Time (Year)	29.5	29.625	29.75	29.875	30
"Population" Runs:	equilib				
Population	1,000	1,000	1,000	1,000	1,000

Comparing Simulations

A key feature of Vensim is the ability to do multiple simulations on a model under different conditions to test the impact that changes in constants (or lookups) have on model behavior. Vensim also stores all the data for all variables for each simulation run, so that you can easily access information about the behavior of any variable in any run. Experiments are performed by *temporarily* changing Constant or Lookup values and then simulating the model. This way, your underlying model stays the same, an unchanging reference point.

Exponential Growth

Now that we are satisfied that we have equilibrium conditions, let us make changes to the model constants to generate unconstrained growth. This is one of the simplest possible dynamic behaviors, known as exponential growth.

Setting Up a Simulation Experiment

- Click on the **Set Up a Simulation** button.

The Toolbar changes to the simulation toolbar.



This toolbar has features specific to model simulation, allowing changes to the integration technique, and buttons to change model constants and lookups. You will also notice that constants in the sketch of this Population model turn into yellow words with blue background. Also, the sketch tools are grayed out, preventing work in the sketch window.

- Click on the **Runname** editing box and replace *equilib* with the name *growth*.
- Click on the variable *birth rate* (appearing blue/yellow in the sketch) and in the editing box type the value 0.2. Press the Enter key. This is a temporary change for this run only and does not permanently alter the value in your model.
- Click the **Simulate** button and the model will simulate.

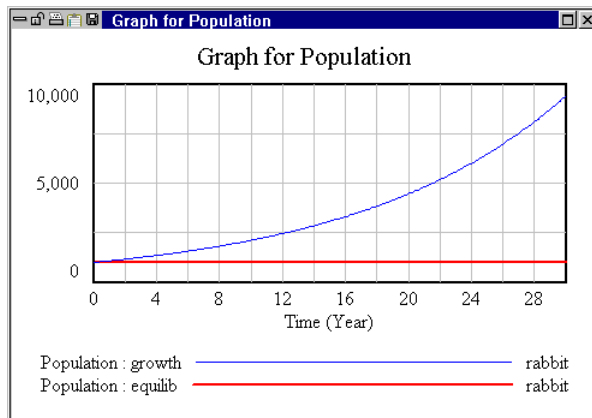
Causes Strip Graph

- Click on the **Control Panel** button to bring the Control Panel to the front. Click the **Datasets** tab to open the Datasets Control and check that both runs are loaded in the right hand column.

The last run you made (*growth*) is loaded first (at the top of the dialog). Most Analysis tools act on both datasets, allowing comparison of behavior from both runs.

- Click on the **Graph** tool.

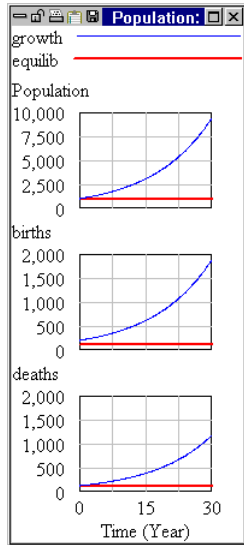
A graph will be generated showing both runs (see below).



- Press the Del key or click the **Close** button to remove the graph.
- Now click on the **Causes Strip** tool.

6: Vensim PLE User's Guide

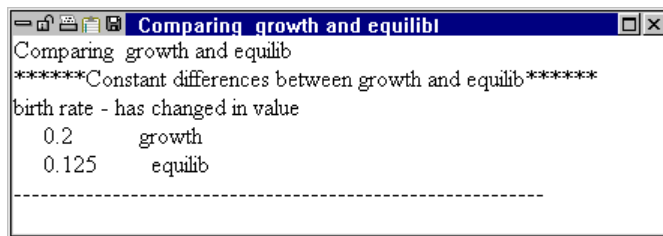
A strip graph is generated showing *Population* and its causes *births* and *deaths*.



Runs Compare

To discover the differences between the first and second runs, we will use a tool that compares all Constant (and Lookup) differences. This tool acts on the first two loaded runs (check in the Datasets Control).

- Click on the **Runs Compare** tool. The text report below shows the differences in the Constant *birth rate* for runs *equilib* and *growth*.



Population grew in the *growth* run because the *birth rate* was set to a higher value than the equilibrium value. This made the positive feedback loop through *births* stronger than the negative feedback loop through *deaths*, resulting in *Population* growth over time.

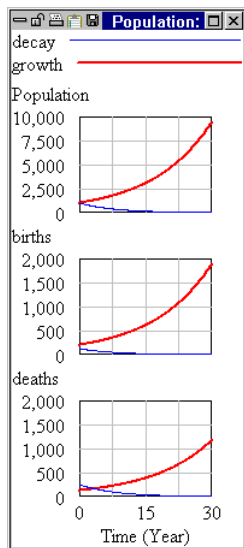
- Select the Menu Item **Windows>Close All Output**.

All of the windows you have created using the Analysis tools will be closed.

Exponential Decay

Next, we will make changes to a model Constant to generate exponential decay or decline in the population. Like exponential growth, this is one of the simplest possible dynamic behaviors.

- ▶ Click on the **Set Up a Simulation** button.
- ▶ Type in the run name *decay* to replace the name *growth*.
- ▶ Click on the variable *average lifetime* (appearing blue/yellow in the sketch) and in the editing box type the value 4. Press the Enter key. This is a temporary change for this run only and does not permanently alter the value in your model.
- ▶ Click the **Simulate** button and the model will simulate.
- ▶ Click on the **Graph** tool and compare the three runs.
- ▶ Click on the **Control Panel** button on the Main Toolbar. In the Datasets Control box, double click on the run *equilib* in the right box; this will unload the run so the Analysis tools will not examine it. Note that it can be reloaded just as easily. (You can also single click to select the run then use the **Move** button (<<) or (>>) to unload or load the dataset.)
- ▶ Click on the **Causes Strip** tool; a strip graph is generated showing *Population* and its causes — *births* and *deaths*, for the two last runs.



Population declines in the *decay* run because the *average lifetime* was set to a lower value than the equilibrium value. This made the negative feedback loop through *deaths* stronger than the positive feedback loop through *births*, resulting in *Population* decline over time.

Input and Output Objects

Another way to control simulations and examine output is by using Input and Output objects. The Input object is a slider which represents some variable in your model for which you can change the value. The Output object is output from an Analysis tool, commonly a graph or a table. See Chapter 11 Input and Output Objects for more details. Input Output Objects are only available in Vensim PLE Plus.

7 Building a Function with Lookups

The population model presented in the previous chapter is a simple model that uses only multiplication and division in its Rate (or Auxiliary) equations. While addition, subtraction, multiplication and division are the most common components of equations, sometimes it is necessary to use different types of relationships. Vensim has a number of functions such as EXP (exponential) and LN (natural logarithm) that can be helpful. Often, however, it is more convenient to create special functions with the properties or shape that you want.

Lookups allow you to define customized relationships between a variable and its causes. Instead of the simple multiplication or division equations:

$$y = a * x \quad \text{or} \quad y = a / x$$

an equation can be defined with a specially-constructed function:

$$y = fn(x)$$

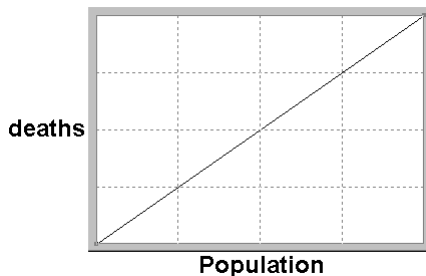
The output variable y is changed by input variable x through the Lookup function fn , which has a particular shape or relationship that is non-linear (in most cases).

Lookups are also known as Lookup Functions, Graphical Functions, Lookup Tables, or simply Tables. They can be constructed as a table of numbers (in the Equation Editor), or as a graph (in the Graph Lookup Editor).

Limits to Rabbit Growth

The population model in the previous chapter is extended here to include a consequence of rapid population growth. The unchecked growth of the previous model (*pop.mdl*) is replaced by growth which is limited by the size of the population (*rabbit.mdl*).

The value of the variable *deaths* in *pop.mdl* (Chapter 6) is directly proportional to the size of the rabbit population. In effect, we see a relationship between *Population* and *deaths* that is linear:

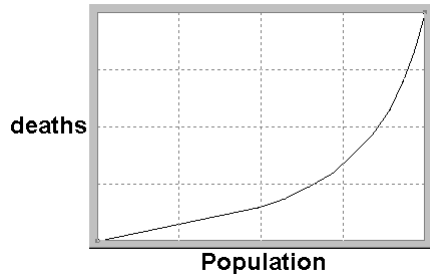


This *does not* mean that *deaths* will increase linearly over time. The linearity means *deaths* will grow at the same rate as *Population* (if *Population* grows exponentially, so will *deaths*).

7: Vensim PLE User's Guide

We could make a Lookup table (as above) that expresses this linear relationship, but it is easier just to use a constant (that has the same value as the slope of the Lookup) multiplied by *Population*. In fact, the model developed in Chapter 6 uses *Population/average lifetime* so that the slope of the straight line would be $1/\text{average lifetime}$. This formulation was chosen because it is much easier to understand what *average lifetime* is (and what a change in it means) than understanding the shape and slope on curves.

What we want to do in this model is have the *deaths* rate increase more quickly than *Population* as population grows to a large size. This happens because higher populations are nearer resource limits (such as food) and therefore die off more quickly. We are looking for a function such as:



You can develop such a function using Lookups, and that is what you will do next. Before that happens, however, it is important to normalize the input and output of the Lookup.

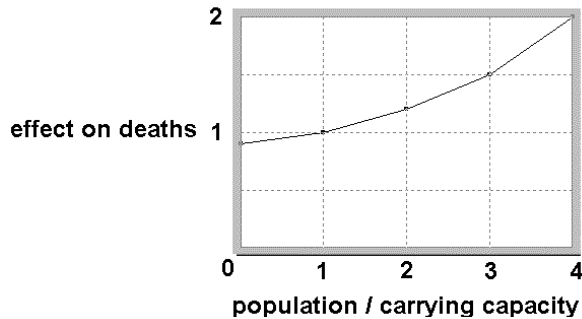
Normalized Lookups

The graphical function drawn above uses an input of quantity of rabbits and outputs the number of rabbits that die per year. This is a difficult graph to create, and worse, it is really tricky to change. Suppose, for example, you want to understand what happens when a longer lived breed of rabbits is introduced — the whole function needs to be redone. Or suppose you want to understand the effects of increasing the carrying capacity of the rabbit's environment — the whole function needs to be redone.

A normalized input is built around reference points such as 0,0 and 1,1. The input is adjusted to be dimensionless and independent of the units of measure of other variables in the model. The output is very often dimensionless and is also independent of the units of measure for other variables. For example, suppose we were to measure *Population* in thousands of rabbits and *deaths* in thousands of rabbits per month. A Lookup function taking *Population* as its input and having *deaths* as its output would no longer be valid.

On the other hand, a Lookup function, normalized by using *Population* relative to *carrying capacity* in its input and having the output *effect on deaths* act on a baseline or normal number of deaths ($\text{Population} / \text{average lifetime}$), does not need to be changed when conditions change.

Normalization allows us to achieve the desired behavioral relationship with a generalized set of numbers in the Lookup function. If information about the size or characteristics of the population changes, you can simply change the value of *carrying capacity* or *average lifetime*, rather than the whole Lookup function.



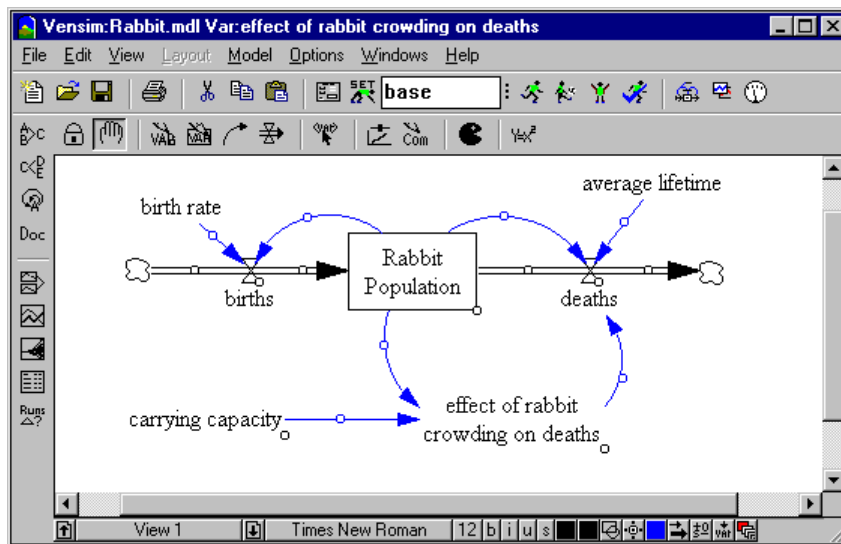
To normalize, divide the input variable by a normal or average value (e.g., *Population / carrying capacity*). When the actual *Population* is equal to this normal value, the input (to the Lookup) is 1. Other values of *Population* will vary the input either higher or lower than 1. The output values of a Lookup are also usually made to vary about the value of 1. The Lookup can then drive another model variable above or below its normal value. Another way of saying this is: when the input variable is equal to its normal or average value, the output from the Lookup is 1 and therefore has no effect on the connected output variable's normal value.

NOTE This formula is similar to the Lookup equation example in the beginning of this chapter $y = fn(x)$, but takes two variables as the (normalized) input and computes the output relative to a normalized value: $y = \text{normal } y * fn(x/\text{normal } x)$.

Sketching the Model (*rabbit.mdl*)

This model (*rabbit.mdl*) demonstrates the simple and direct approach to building models with Lookup functions. The normalization is done *within* the Lookup output variable *effect of rabbit crowding on deaths* and the output of this variable acts directly on the rate *deaths*. To see an example where the input and outputs are separated, see the section Separate Normalized Variables later in this chapter, or open the model *rabbit2.mdl*. This is functionally the same model as *rabbit.mdl* but contains more variables. You decide which is clearer.

- ▶ Click the **New Model** button.
- ▶ In the Model Settings dialog, Time Bounds tab, type 30 for FINAL TIME, type (or select from the drop down box) 0.125 for TIME STEP. Click on the drop-down box for **Units for Time**, and select **Year** (or type in Year).
- ▶ Click the **Units Equiv** tab. In the editing box, type *rabbit , rabbits* then click the button **Add Editing**. Click on **OK** (or press Enter).
- ▶ Sketch the model shown below in the diagram below.
- ▶ Save your model (e.g., *rabbit.mdl*) in the *plemodel\chap07* directory.



Entering Equations

- Click the **Equations** tool, click on the following variables and enter the equations and units of measurements as follows (see Built-in Functions section below for *birth rate* equation):

```
Rabbit Population = INTEG (
    births-deaths,
    initial population)
```

```
Units: rabbit
```

```
initial population =
    1000
```

```
Units: rabbit
```

```
birth rate=
    0.23
```

```
Units: fraction/Year
```

```
average lifetime =
    8
```

```
Units: Year
```

```
births =
    Rabbit Population * birth rate
```

```
Units: rabbit/Year
```

```
deaths =
    (Rabbit Population / average lifetime) * effect of rabbit
    crowding on deaths
```

```
Units: rabbit/Year
```

```
carrying capacity =
  1000
Units: rabbit
```

The variable *carrying capacity* is *not* the maximum amount of rabbits that the environment can hold. Instead, *carrying capacity* represents the *normal* amount of rabbits for that environment. We would use a different formulation if we wanted to represent maximum carrying capacity.

Note that we are initializing the Level with the Constant *initial population* rather than typing in a number. This allows us to make changes to the value during simulation experiments. Note also that the *deaths* equation has a multiplier *effect of rabbit crowding on deaths* which will change the value of deaths.

Creating and Normalizing Lookups

- ▶ With the **Equations** tool selected, click on *effect of rabbit crowding on deaths*.

Under the **Type** label there are two drop down boxes, one showing Auxiliary, the other showing Normal.

- ▶ Click on the drop down box showing Normal and select **with Lookup**.

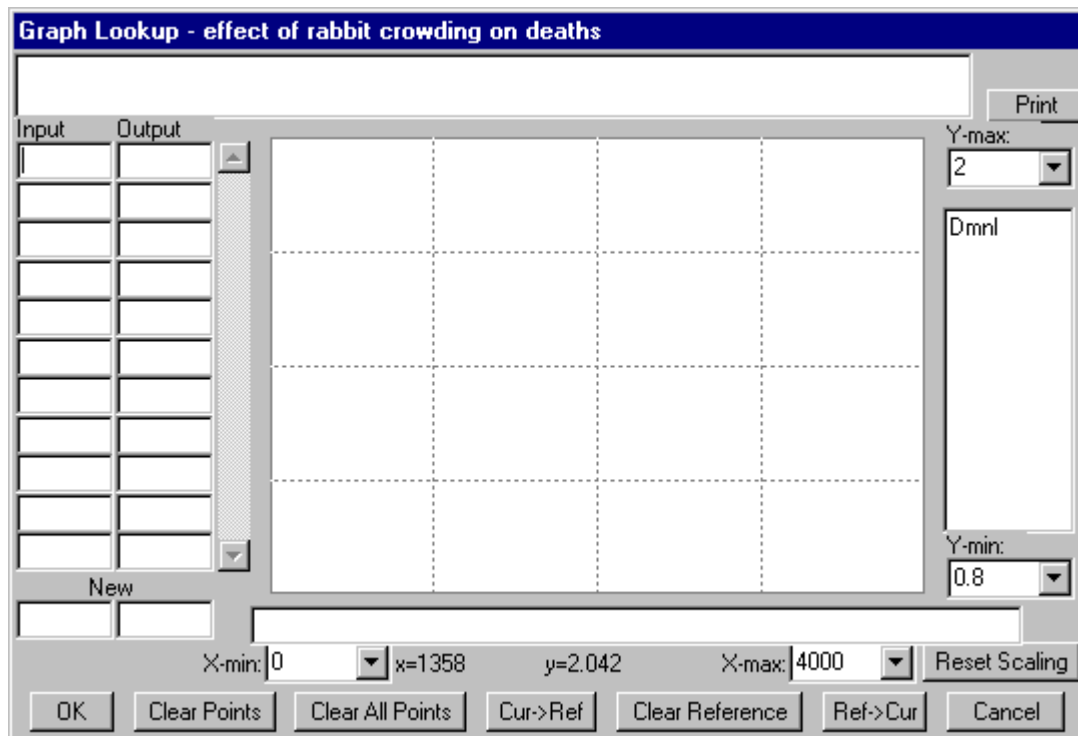
This creates a Lookup table embedded within the Auxiliary variable.

- ▶ Click on the **Variables** tab then click on the variable *Rabbit Population* in the list.
- ▶ Type a divide by symbol (/) then click on *carrying capacity*.

```
effect of rabbit crowding on deaths = WITH LOOKUP(
  Rabbit Population/carrying capacity
```

As *Rabbit Population* changes, the *effect of rabbit crowding on deaths* will change according to the shape of the Lookup function. Now we need to create the actual table of figures or graph that describes the Lookup.

- ▶ Click on the button **As Graph** in the Equation Editor (below the **Type** boxes). The Graph Lookup Editor opens:

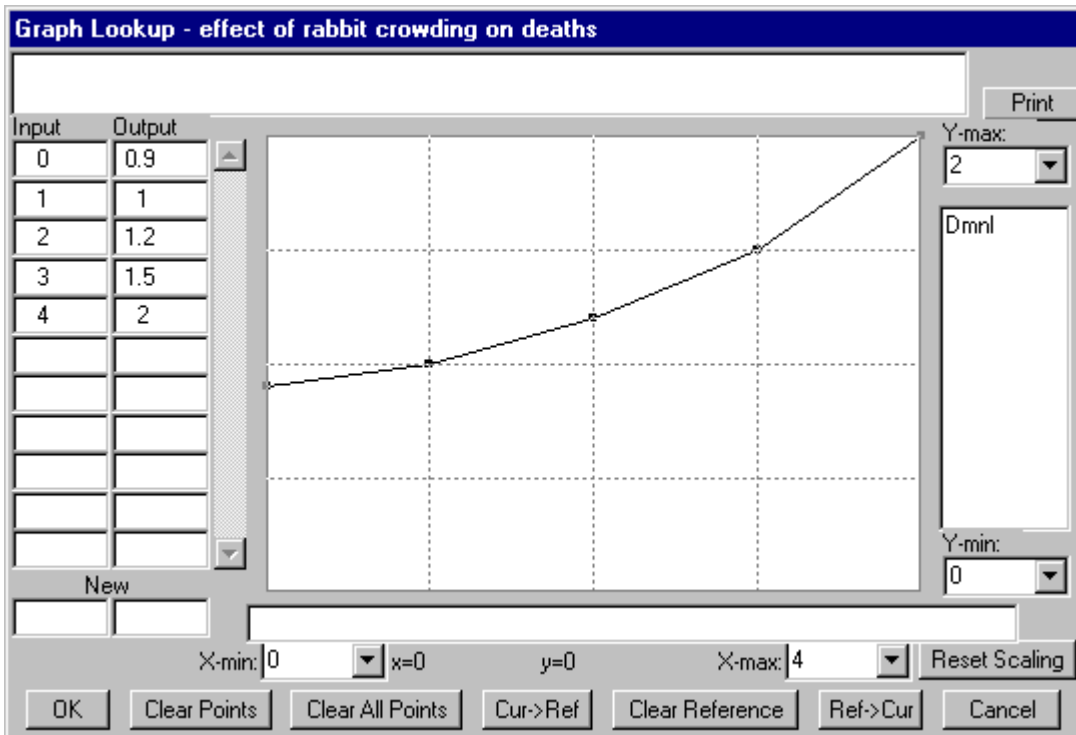


- ▶ Click on the left hand **New** (values) box near the bottom left corner and type in 0 then press the Enter key. The cursor moves to the right hand box; type 0.9 and press Enter again. The cursor moves back to the left hand box and the old numbers are moved to the input/output columns making way for new numbers to be entered.
- ▶ Continue typing in the rest of the pairs of values below, pressing the Enter key each time a value is typed. The graph will draw itself.
 $(0, 0.9), (1, 1), (2, 1.2), (3, 1.5), (4, 2)$
- ▶ Alternatively, you can sketch the graph using the pointer by clicking on the graph to add points, and dragging the points to reposition them on the graph. You will first need to set the **X-max** and **Y-max** values to 4 and 2 respectively.
- ▶ Click on the button **Reset Scaling** to set the X and Y scales to your points.

Do not worry about getting the exact numerical values for points shown in the figure if you are adding points with the mouse. The shape of the curve is more important than the exact values. But do set (1,1) exactly because this is a reference point: when *Rabbit Population* is equal to *carrying capacity*, there is no change on the normal rate of *deaths*.

Editing Values

- ▶ You can modify values in the **Input/Output** list, or by dragging a point around on the graph.
- ▶ To remove a point, click on the button **Clear Points** and then with the **Delete** icon, click on the point on the graph. Your graph should look like:



- ▶ Click on **OK** and the Graph Lookup Editor closes.

Now you will see the graph equation expressed as a table of values enclosed in parentheses. These values could have been typed in directly, but instead we generated them in the Graph Lookup Editor.

- ▶ Add the units *Dmnl* (for Dimensionless, which you could also have typed in) to the **Units** editing box, then click **OK** to close the Equation Editor.

The units Dimensionless are pretty important to understand. When we normalized the input to the Lookup, we divided the *Rabbit Population* (measured in rabbits) by *carrying capacity* (also measured in rabbits) leaving a dimensionless variable.

- ▶ Click on the **Save** button on the Toolbar to save your model.

Checking for Model Syntax and Units Errors

Before you simulate the model, you should check it for errors in equations and units.

- ▶ Select **Model>Check Model** (or press Ctrl+T), you should get an information box saying "Model is OK."

If the model has errors, check that the structure is the same as in diagram. If the structure looks the same, open the Equation Editor on each variable and check the equation against the list above.

- ▶ Select **Model>Units Check** from the menu (or press Ctrl + U); you should get an information box saying "Units are AOK."

If a units error is generated, read the Output window to see which variables are failing the check. Open the Equation Editor on each variable and check the units against the equations listed above. Units not checking out is often a good indication of poor or incorrect equation formulation.

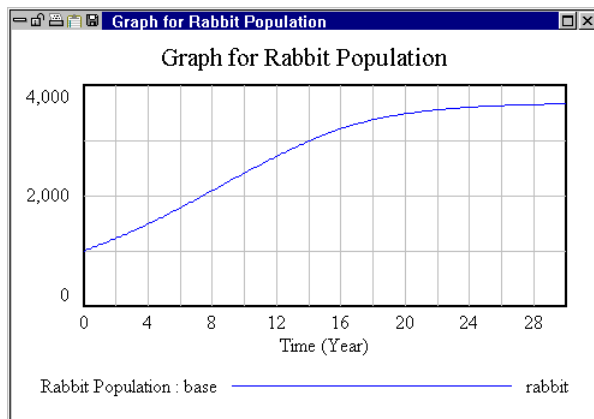
Simulating the Model

- ▶ Click on the **Runname** editing box on the Toolbar and type a name for the first run (e.g., *base*).
- ▶ Click on the **Simulate** button.

The model will simulate, showing a work-in-progress window until completion (on a fast computer, you might not see this window).

Model Analysis

- ▶ Double click on the Level *Rabbit Population* in the sketch. This selects it as the Workbench Variable. Check the title bar at the top of the Vensim window to see that *Rabbit Population* is selected.
- ▶ Click on the **Graph** tool. A graph of *Rabbit Population* is generated:

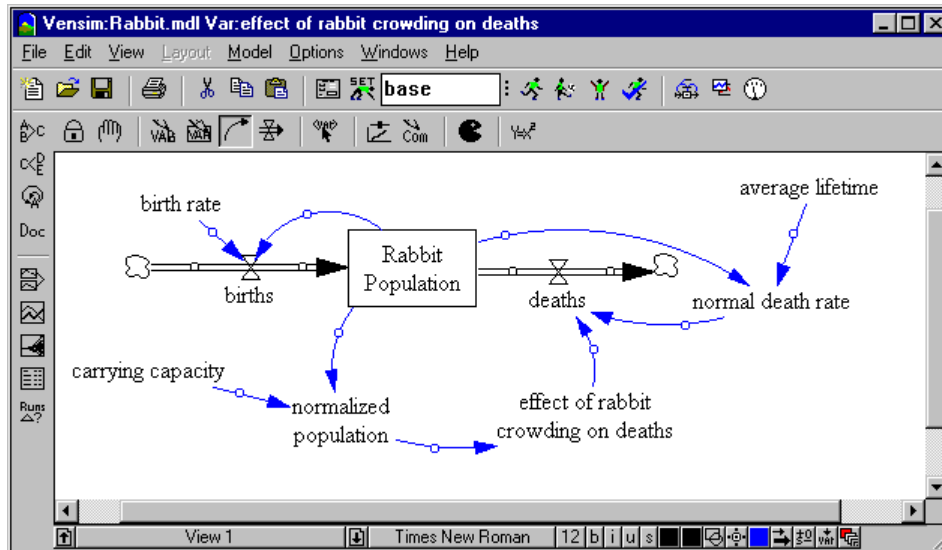


We can see that the *Rabbit Population* first grows exponentially then this growth slows down till *Rabbit Population* approaches a maximum value of approximately 3500 (because of the effect of the Lookup table).

Separate Normalized Variables

This section is optional and produces a model with the same behavior as the *rabbit.mdl* model already constructed.

Lookup tables are best used to drive model variables above or below their "normal" values. In the previous model (*rabbit.mdl*), the normal value was implicit and hidden within the Rate equation for deaths. In other models, you might want to have an explicit normal value. This might be a Constant, or an Auxiliary as *normal death rate* in the *rabbit2.mdl* model below:



This model also includes an explicit (separated) normalized input, rather than normalizing the input variable inside the variable *effect of rabbit crowding on deaths*.

- ▶ Save the previous rabbit model as another name (*rabbit2.mdl*) and then create the structure above.
- ▶ To move the arrows from the Rate *deaths* to the Auxiliary *normal death rate*, use the **Move/Size** tool to grab the arrowhead from *deaths* and drop it on the Auxiliary. Alternatively, you can delete the arrows using the **Delete** tool by clicking on the arrowhead, then draw new arrows with the **Arrow** tool.
- ▶ Click on *normal death rate* and enter the following equation (the same equation for deaths before we introduced the effect from the Lookup) and units then press Enter:

```
normal death rate=
  Rabbit Population / average lifetime
```

7: Vensim PLE User's Guide

Units: *rabbit/Year*

- Click on *deaths* and replace the equation with the one below then press Enter.

deaths=

normal death rate * *effect of rabbit crowding on deaths*

- Click on *normalized population* and enter the equation below:

normalized population =

Rabbit Population / *carrying capacity*

Units: *Dmnl*

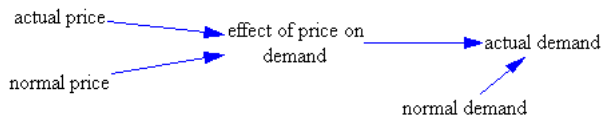
- Click on *effect of rabbit crowding on deaths* and replace the equation in the WITH LOOKUP editing box with (leaving the Lookup table and units as they are):

effect of rabbit crowding on deaths= WITH LOOKUP (
normalized population

The Equation Editor should close and no variables on the diagram should have highlights.

- Click on the **Save** button on the Toolbar to save your model.

Note that the normal output variable *normal death rate* could (in a different model) be a Constant, and not the Auxiliary that it is here. See for example, the structure below:



Simulation

- Before simulating, perform a Units Check (Ctrl + U) and a Model Check (Ctrl + T).
- Choose a simulation run name and simulate the model. Check output with the graph tools. You should get exactly the same behavior as the previous model (see graph earlier).

One thing to note is the extra variable *normal death rate*. This calculates what would be the amount of deaths if there was no population crowding affecting the death rate (via the Lookup table). We could have embedded this calculation in the Rate *deaths* as in the previous model (rabbit), but we chose to separate it for clarity and to teach the use of Lookup output driving a "normalized" variable.

Changing Model Lookups

Now let us make a temporary change to the Lookup and simulate the model again.

- Click on the **Set Up a Simulation** button.
- Click on the **Runname** editing box and type *run2* or some other run name.

7: Building a Function with Lookups

- Click on the variable *effect of rabbit crowding on deaths* appearing yellow/blue in the sketch.

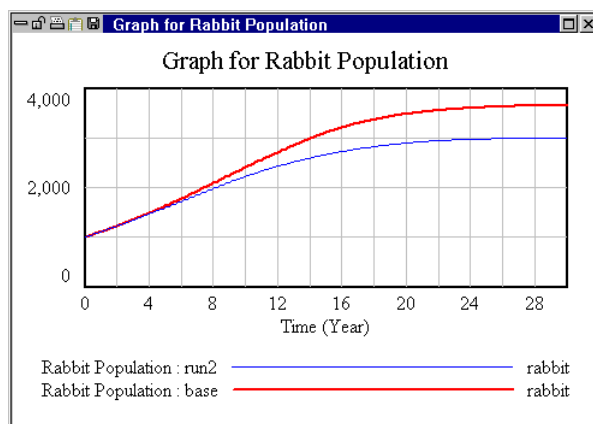
The Graph Lookup Editor will appear.

- With the mouse, move some of the points on the graph to change the steepness of the curve. For example, drag the points a little higher. If you need to, increase or decrease the scale by clicking on the drop down boxes for **Y-max:** **Y-min:** **X-max:** **X-min:** or typing in new values. You can also add extra points or delete points. (You could also change the values from the keyboard in the **Input** and **Output** fields on the left.) Click the **OK** button.

This is a *temporary* change for this run only, and does not permanently alter the values in your model.

- Click the **Simulate** button and the model will simulate.
- Double click on *Rabbit Population* to select it, then click on the **Graph** tool.

You might get a graph such as the one below, showing reduced size in final *Rabbit Population*, or something quite different. The graph below was a result of an increased effect of the Lookup table (increased values).



Named Lookups

In the previous models we have used an Auxiliary variable with subtype with Lookup to enter the nonlinear effect of population density on the rate at which rabbits die. In some cases it is desirable to place a name on the functional form. This is especially true if you want to use the Lookup in more than one place.

- Save the model *rabbit2.mdl* as a new name (*rabbit3.mdl*).
- Click on the Variable tool and add the new variable *effect of rabbit crowding on deaths function*.
- Draw an arrow from the *effect of rabbit crowding on deaths function* to *effect of rabbit crowding on deaths*.

7: Vensim PLE User's Guide

- Open the equation editor on *effect of rabbit crowding on deaths function*.
- Click on the dropdown for type and select Lookup. Click on the **As Graph** button to open the Graph Editor and enter the Lookup values as you did before.
- Close the Graph Editor, add in units Dmnl and click on **OK** to close the Equation Editor. Your equation should look like:

```
effect of rabbit crowding on deaths function(  
  [(0,0)-(4,2)],(0,0.9),(1,1),(2,1.2),(3,1.5),(4,2))  
Units: Dmnl
```

- Open the equation editor on *effect of rabbit crowding on deaths*.
- Select the Subtype **Normal** from the dropdown. The lower window will disappear. Click on the beginning of the equation and select *effect of rabbit crowding on deaths function* from the **Variable** list. Add parentheses () around *normalized population*.

```
effect of rabbit crowding on deaths=  
  effect of rabbit crowding on deaths function(  
    normalized population)
```

- Click on **OK** to close the dialog.

This model is exactly the same as *rabbit2.mdl* except that the Lookup relationship has been explicitly named. While naming Lookups in this manner is cumbersome for a simple relationship such as the one defined here it can be very helpful in more complex situations, especially if you want to use the same nonlinear relationship in more than one place. When you click on the Simulation Setup button the named Lookup will highlight allowing you to change it.

8

Multiple Views

How Views Work

Large models can be presented more clearly in multiple views than in a single view. Views can be thought of as similar to pages of a book, each page telling a portion of the story. Each view displays a sketch and is connected to one (or more) of the other views through variables or shadow variables. Multiple views allow models to be broken into sectors, such as production, financial, customer and so on.

Only Vensim PLE Plus can create multiple views. If you are using Vensim PLE, you may still want to work through this chapter, but you will need to make all your changes in a single view. Alternate instructions are included for working with Vensim PLE.

Vensim PLE will open models with multiple views. You can move through the views using the Page Up and Page Down keys, or by clicking on the name of the View in the StatusBar.

Graphic Models and Variables

Vensim models are definitively expressed in equation or text form. The variable can be shown as a regular variable (with causes attached), or a shadow variable (with no causes attached).

NOTE It is possible to build separate working models in different views, but this is not recommended unless you intend to link them causally together at some later time. It is better to build complete new models (this gives them their own name and Time Bounds).

Customer Diffusion Model

This model describes a simple diffusion process, where Potential Customers of a product are influenced into buying the product by word of mouth from Customers (who already own the product). The first view shows the diffusion process. The second view will add a production capacity variable which will potentially limit the quantity of product sold at any one time. The third view will describe the sales revenue generated from sales of the product.

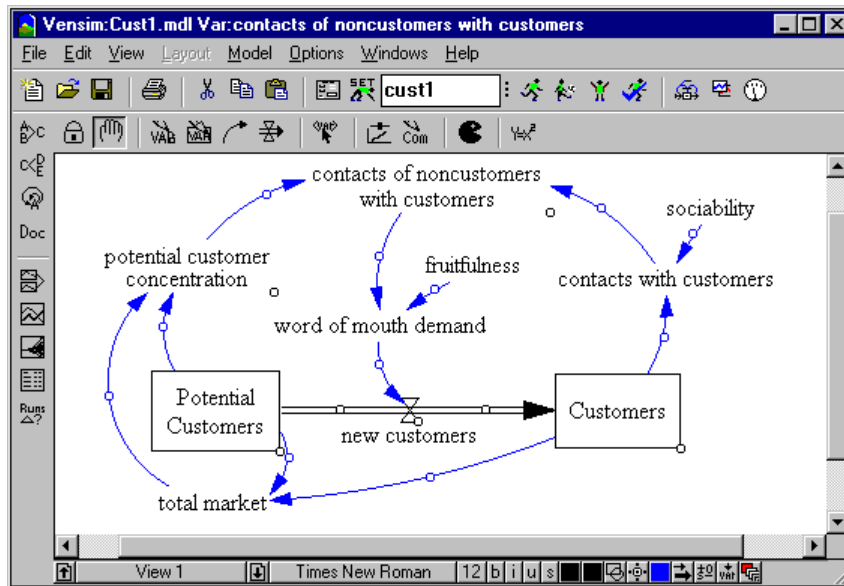
Building the Diffusion Model (*cust1.mdl*)

EITHER

- Open the existing model: Click the **Open Model** button and select the model *cust1.mdl* from the *plemodel\chap08\complete* directory and save it as *cust1.mdl* in the *plemodel\chap08* directory, then skip forward to section "Simulating the Model."

OR

- Build a new model: Click the **New Model** button or select the menu item **File>New Model**
- Click **OK** to accept the default Time Bounds.
- Sketch the model shown in the figure below.
- Save your model as *cust1.mdl* in the *plemodel\chap08* directory.



Entering Equations

- Click on the **Equations** tool and enter the following equations and units of measurement for each variable in the model:

new customers = *word of mouth demand*

Units: *person/Month*

contacts of noncustomers with customers = *contacts with customers* * *potential customer concentration*

Units: *contacts/Month*

```

contacts with customers = Customers * sociability
Units: contacts/Month

Customers = INTEG(
    new customers,
    1000)
Units: person

fruitfulness = 0.01
Units: person/contacts

potential customer concentration = Potential Customers / total market
Units: dmnl

Potential Customers = INTEG(
    - new customers,
    1e+006)
Units: person

sociability = 20
Units: contacts/person/Month

total market = Customers + Potential Customers
Units: person

word of mouth demand = contacts of noncustomers with customers *
    fruitfulness
Units: person/Month

```

Checking for Model Syntax and Units Errors

Before you simulate the model, you should check it for errors in formulas and units.

- ▶ Select **Model>Check Model** (or Press Ctrl + T), you should get an information box saying "Model is OK."

If the model has errors, check that the structure is the same as in diagram . If the structure looks the same, open the Equation Editor on each variable and check the formula against the list above.

- ▶ Select **Model>Units Check** from the menu (or press Ctrl + U); you should get an information box saying "Units are AOK."

If a units error is generated, read the Output window to see which variables are failing the check. Open the Equation Editor on each variable and check the units against the list above.

- ▶ Save the model by clicking the **Save** button, or selecting menu **File>Save**, or pressing Ctrl + S.

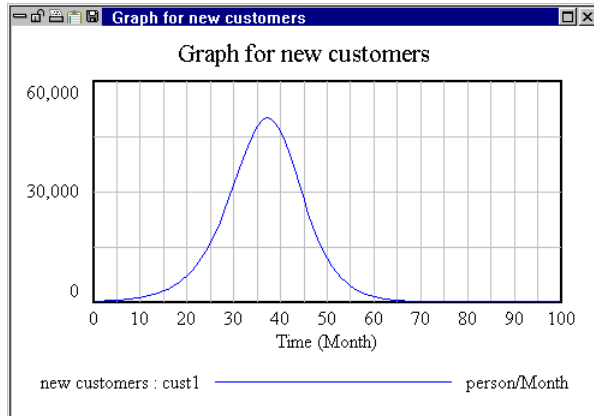
Simulating the Model

- ▶ Double click on the **Runname** editing box and type *cust1* for the first dataset.
- ▶ Click on the **Simulate** button.

Model Analysis

- ▶ Click on the **Graph** tool or the **Causes Strip** graph to investigate the behavior of key variables, such as *Potential Customers*, *Customers*, and *new customers* (as well as any other variables that you want to see).

The variable *new customers* should give you the following graph:



Naming and Saving Your Model

Before starting the next section, name the view, and save the model under a new name so that we will have two distinct working models.

- ▶ (PLE Plus Only) Select menu **View>Rename** and type in the name *Customer*, then click **OK**.
- ▶ Select **File>Save As...** and enter the model name *cust2.mdl*, then click **Save**.

Adding the Capacity View (*cust2.mdl*)

Now we would like to expand our model by considering production capacity issues. Customer demand might outstrip our capacity to supply the product. Therefore, we will build a production capacity view that is linked to the customer view and limits the customer demand if capacity is reached.

In PLE Plus:

- ▶ Select the menu item **View>New**. A new view opens.
- ▶ Select the menu item **View>Rename**. Type in the name *Capacity* and click **OK**.

In PLE:

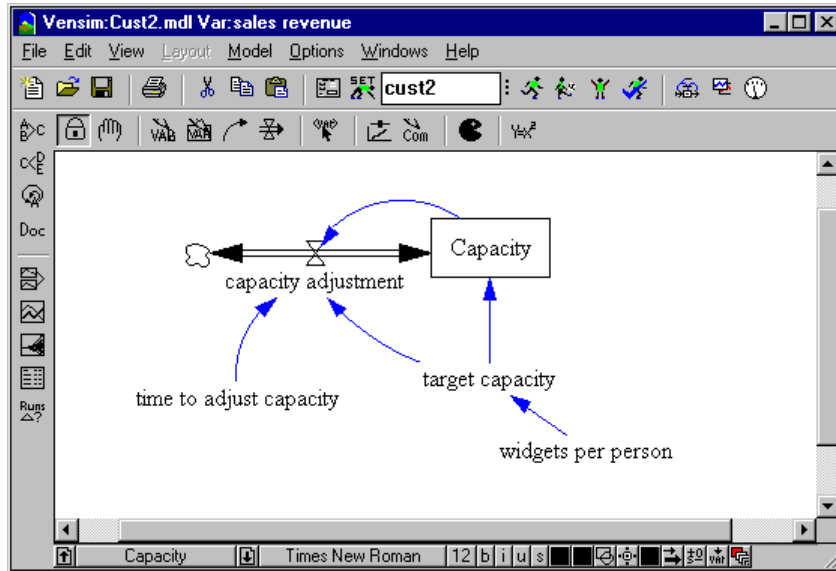
- ▶ Scroll to an empty portion of the current model. If you want to scroll further drag an variable in the direction you want to scroll and the scroll range will automatically increase.

Then:

- ▶ Select the **Box Variable** tool and click on the sketch. Type the name *Capacity* and press Enter.
- ▶ Select the **Rate** tool, then click on the sketch to the left of *Capacity* then click on *Capacity*. Type the name *capacity adjustment* and press Enter.
- ▶ Using the **Move/Size** tool, click with the right mouse button on the handle (the little circle) in the middle of the left hand Rate pipe. The Arrow Options dialog will open. Check the box for **Arrowhead** at the top of the dialog then click **OK**.

This extra arrowhead indicates that this Rate pipe can flow in either direction. Or in other words, the Rate can increase or decrease the Level *Capacity*. Note that actual behavior of the Rate is governed by the equation, not by the arrowhead that we just added.

- ▶ Create the other variables and connect them with arrows as shown below:



Shadow Variable Tool

Select the **Shadow Variable** tool and click on the sketch below *target capacity*. A dialog box opens:

- ▶ Either select the variable *word of mouth demand* from the list, or type the first few letters of *word of mouth demand* until it becomes highlighted in the list, then press Enter or click on **OK**.
- ▶ Select the **Arrow** tool and click on the Shadow Variable *word of mouth demand* then click on *target capacity*.

The structure of this view is now complete. The structure is causally linked to the first view through the variable *word of mouth demand*.

NOTE The shadow variable is only inserted for use in causing *other* things to change. If you try to connect an arrow from another variable to the shadow variable, the arrow will not connect.

Adding Equations

- Select the **Equations** tool.

All variables will appear black, except for the Shadow Variable *word of mouth demand* which has an equation defined in the first view.

- Enter the following equations and units of measurement for each variable in this view:

```
Capacity = INTEG(  
    capacity adjustment,  
    target capacity)  
Units: Widget/Month  
  
capacity adjustment=  
    (target capacity - Capacity) / time to adjust capacity  
Units: Widget/Month/Month  
  
target capacity=  
    word of mouth demand * widgets per person  
Units: Widget/Month  
  
time to adjust capacity=  
    12  
Units: Month  
  
widgets per person=  
    1  
Units: Widget/person
```

Now we will return to the first view and complete the feedback loop by linking its structure to the variables in the view Capacity.

In PLE Plus:

- Click on the **View** button (on the Status Bar — currently reads "Capacity") and choose Customer.

NOTE You can also use the Page Up / Page Down keys on your keyboard to step through views.

In PLE:

- Scroll back to the structure related to customers.

Then:

- Select the **Shadow Variable** tool and click below and right of the variable *new customers*. Choose *Capacity* from the list (or type the first few letters of *Capacity*) and press Enter (or click **OK**).

- ▶ Repeat this process for the variable *widgets per person*.
- ▶ If necessary, move the variable *total market* lower and move the arrows to make room.
- ▶ Select the **Arrow** tool and connect *Capacity to new customers*, then connect *widgets per person* to *new customers*.

Altering An Equation

- ▶ Select the **Equations** tool.

Only the equation for *new customers* should be highlighted black, because this is the only variable in this view to which we added causes. Sales of the product are limited by the factory's ability to produce, so we will write an equation which returns the smaller value of *word of mouth demand* or factory capacity divided by units of product per customer (*Capacity/widgets per person*).

- ▶ Click on *new customers* and change the equation to the one below:

```
new customers=
  MIN(word of mouth demand, Capacity/widgets per person)
```

You can select the MIN function from the list under the **Functions** tab, or just type it in as above.

Adding a Sales Revenue View

Let us add a view which will track the sales revenue from selling the product, and also the cumulative revenue from all the sales.

In PLE Plus:

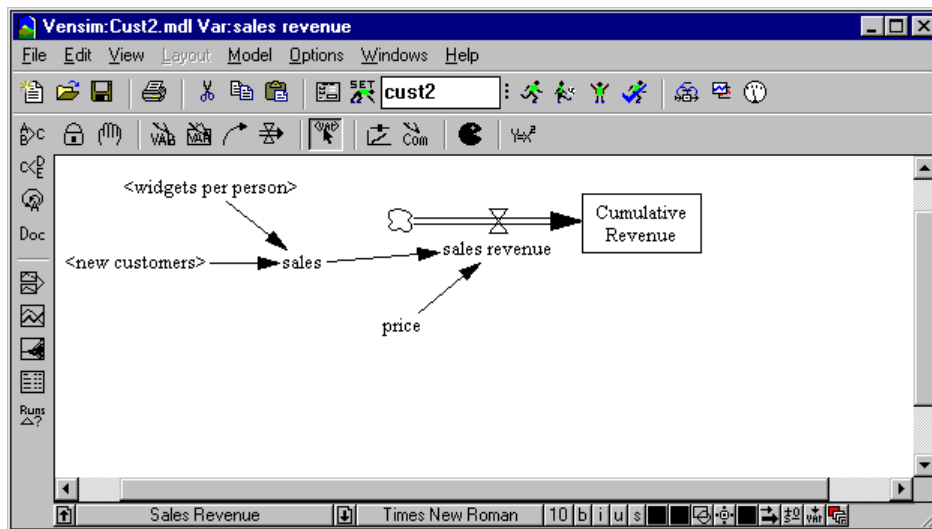
- ▶ Select the menu item **View>New**.
- ▶ Select the menu item **View>Rename** and type in Sales Revenue, then press Enter.

In PLE:

- ▶ Scroll to an empty area of the view.

Then:

- ▶ Select the **Shadow Variable** tool. Click on the view and select the variable *new customers* then press Enter.
- ▶ Click on the view and select the variable *widgets per person* then press enter.
- ▶ Add the variables and arrows as shown in the diagram below:



More Equations

- Click on the **Equations** tool and enter the following equations and units of measurement for each variable that appears black in the view (the others already have equations entered).

```
sales=
    new customers * widgets per person
```

```
Units: widget/Month
```

```
price=
```

```
50
```

```
Units: dollar/widget
```

```
sales revenue=
```

```
sales * price
```

```
Units: dollar/Month
```

- When you click on *Cumulative Revenue* to write its equation, click on the check box marked **Supplementary**. This tells Vensim that the variable is not used anywhere else.

```
Cumulative Revenue = INTEG(
    sales revenue,
    0)
```

```
Units: dollar
```

- Select **File>Save** (or press Ctrl + S).

Simulating the Model

- Check the model for errors with **Model>Check Model** (Ctrl + T).

- ▶ Check the units for consistency by selecting **Model>Units Check** (Ctrl + U).
- ▶ Simulate the model using the dataset name *cust2*.

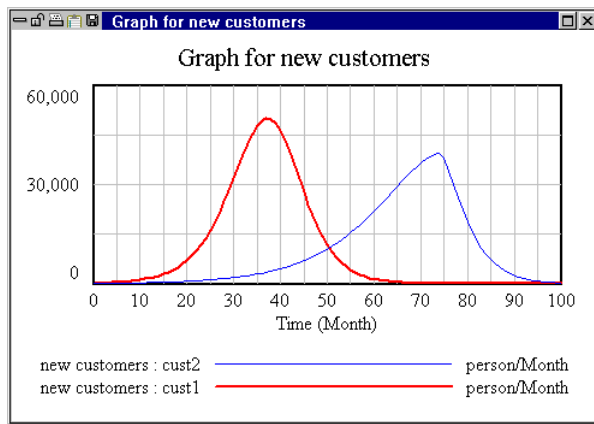
Analyzing the Model

- ▶ Use the **Graph** tool or the **Causes Strip** graph to investigate behavior of key variables, such as *Potential Customers*, *Customers*, and *new customers* (as well as any other variables that you want to see).

If you opened the existing *cust2.mdl* rather than creating your own, you will notice that the only results displayed are those for the current dataset (*cust2*). Use the Control Panel **Datasets** tab to load *cust1*.

- ▶ Select the variable *new customers* as the Workbench Variable and click on the **Graph** tool.

NOTE *cust1* has stored results for all the variables in *cust1.mdl* and does not have any results for the variables unique to our new model. If you ask to see a graph for a variable not in the original model you will only see results from the run *cust2*.



We notice a difference between the two runs. Because of the constraint in production capacity, the second run shows a much slower rise in sales and a later peak, and the total time over which sales are made is longer.

Saving Your Model

Before we start the next section, save the model under a new name so that we have three distinct working models.

- ▶ Select **File>Save As...** and enter the model name *cust3.mdl* then click **OK**.
- ▶ Select **File>Close** to close the model.

Detailed Capacity Model (*cust3.mdl*)

Suppose we want to model more accurately the production capacity sector of a firm. If, somewhere, we have an existing model that describes the capacity sector better than our current model, we can use this model to enhance and refine our current model. We have a model called *cap1.mdl* that builds capacity based on investment with a construction delay, and that depreciates capacity only after the capacity life is used up. This implies that we cannot reduce capacity any faster than it depreciates (unlike our other model where capacity could be reduced at the same Rate as it was built). This section introduces the concept of merging two different models (structure and equations) to form one complete working model.

Copying and Pasting

We are going to make use of model structure from another existing model (*cap1.mdl*). This model has an alternative, and somewhat improved, set of equations for determining capacity. It does not, however, have any equations around customer demand. Instead it uses a built in function called STEP to determine *desired production*. (See Chapter 10 for some discussion of the STEP function). We want to paste this structure into our model and then make the connections from variables we have already defined to the variables from *cap1.mdl*.

- ▶ Click the **Open Model** button and open the model *cap1.mdl* in the directory *plomodel\chap08*.
- ▶ Select the menu item **Edit>Select all** (or Ctrl + A, or using the **Move/Size** tool, drag a box to cover all the structure).
- ▶ Click the **Copy** button, or select menu item **Edit>Copy** (or Ctrl + C).
- ▶ Click the **Open Model** button, or select menu item **File>Open Model...** and open the model *cust3.mdl* (or select *cust3.mdl* from the **File** menu recent files list).

In PLE Plus:

- ▶ If you are not in the Capacity view, click on the **View** button and choose the view Capacity.

In PLE:

- ▶ Scroll to the structure around Capacity.

Then:

- ▶ Select the Delete tool and delete all the variables currently associated with capacity except for *widgets per person* and *word of mouth demand*.
- ▶ Click the **Paste** button, or select menu item **Edit>Paste** (or Ctrl + V).

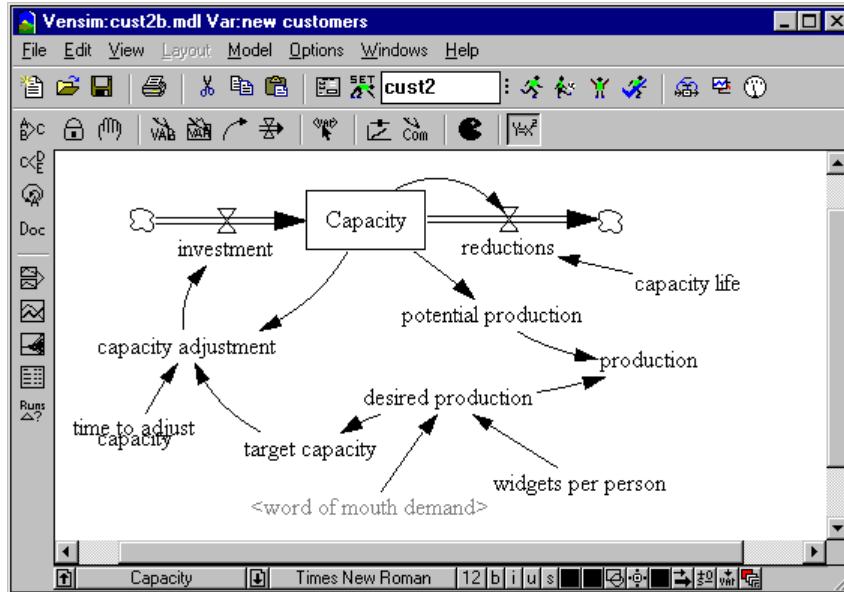
The new structure is dropped onto the sketch view in the same location as it was copied from.

- ▶ Position the **Move/Size** tool in the middle of the highlighted box then press and hold down the mouse button and drag the box to a position that does not overlap the remaining variables.
- ▶ Click outside the box to drop the variables permanently on the sketch.

NOTE If you click outside of the box before you have moved it, the new structure will drop on top of the existing structure; not what we want at this stage. If you have clicked outside the box before moving it: Select **Edit>Undo** and the pasted structure will disappear. Then paste again.

- ▶ Draw an arrow from *word of mouth demand* to *desired production*.
- ▶ Draw an arrow from *widgets per person* to *desired production*.

Your sketch should look something like this:



Now we will fix the equations.

- ▶ Select the **Equations** tool and open the Equation Editor on *desired production*.
- ▶ Replace the existing equation with:

desired production =

$$\text{word of mouth demand} * \text{widgets per person}$$
Units: Gadget/Month

In Vensim PLE Plus:

- ▶ Click on the **View** button and choose the view Customers.

In PLE:

- ▶ Scroll to the structure dealing with customers.

Then:

- ▶ Select the **Shadow Variable** tool and add *production*.
- ▶ Draw an arrow from *production* to *new customers*.

8: Vensim PLE User's Guide

- Open the Equation Editor on *new customers* and change the existing equation to:

```
new customers =  
    MIN(word of mouth demand, production/widgets per person)
```

- Check the model with the command **Model>Check Model** or Ctrl+T.

If the model is not OK check the equations against those listed below.

Capacity View Equations

```
Capacity= INTEG (  
    investment-reductions,  
    target capacity )  
Units: Gadget/Month  
capacity adjustment=  
    (target capacity - Capacity)/time to adjust capacity  
Units: Gadget/Month/Month  
capacity life=  
    20  
Units: Month  
target capacity =  
    desired production  
    Units: Gadget/Month  
investment=  
    capacity adjustment  
Units: Gadget/Month/Month  
potential production=  
    Capacity  
Units: Gadget/Month  
production=  
    MIN(desired production, potential production)  
Units: Gadget/Month  
reductions=  
    Capacity/capacity life  
Units: Gadget/(Month*Month)  
desired production=  
    word of mouth demand * widgets per person  
Units: Gadget/Month  
time to adjust capacity=12  
Units: Month
```

Units Synonyms

Units Synonyms in Vensim are different names that refer to the same unit of measurement. When typing in units, you might find yourself making a unit of measurement singular in one place and plural in another. The Units Check feature will consider them different units unless told that they are synonyms.

Some synonyms are already defined; Month and Months, Year and Years, etc.

- Select **Model>Units Check** from the menu (or press Ctrl + U).

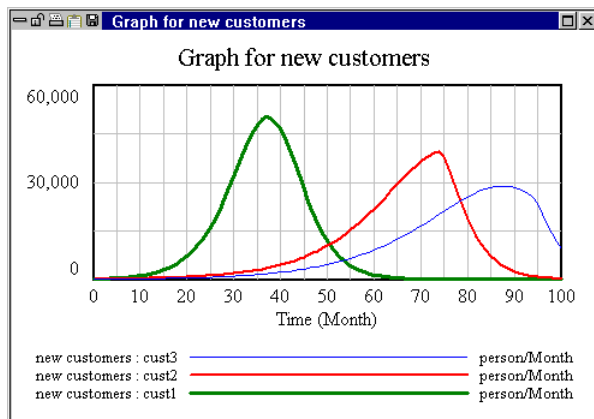
You should get 2 units errors. These are all due to the presence of different names for the things that get produced and sold — Widgets and Gadgets. We could go through each equation looking for Gadget and replacing it with Widget (or vice versa). However, Gadget and Widget are referring to the same unit and we can therefore define them as synonyms. We will also add the plural forms (Gadgets, Widgets) in case we typed a plural by mistake.

- Select the menu item **Model>Settings...** and click the tab **Units Equiv.**
- In the editing box, type Gadget, Gadgets, Widget, Widgets and then click on the button **Add Editing**, then on **OK**.
- Select **Model>Units Check** from the menu (or press Ctrl + U).

Now units should check out AOK. If they do not, read the units errors Output window and try to figure out why. Refer to the units for each variable in the equation set above.

Simulating and Analyzing the Model

- Simulate the model using the dataset name *cust3*.
- Click on the **Control Panel** button then on the **Datasets** tab. Load the datasets *cust1* and *cust2* if not already loaded.
- Select the variable *new customers* as the Workbench Variable and click on the **Graph** tool.

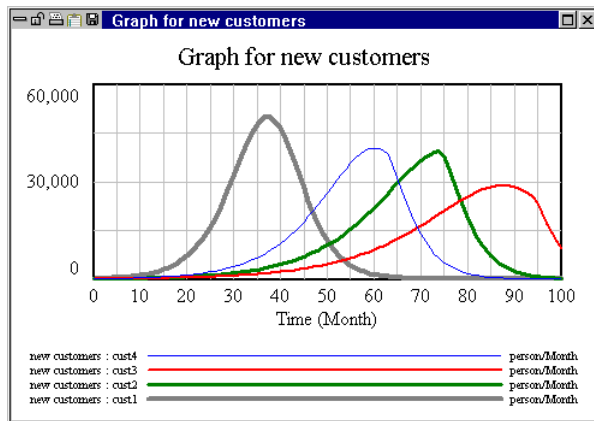


Note how the curve for *new customers* in *cust3* builds more slowly and peaks later and lasts longer than for the other runs. This behavior comes from the constraint in *Capacity*, which is lower in *cust3.mdl* than *cust2.mdl* because of capacity reductions (retirement due to old age).

Capacity Investment Policy

Let us say that we stand to lose market share (*new customers* and *Customers*) because we are supplying the product later than some competitor. What can we do to supply the product (and hence bring on *new customers*) sooner? Let us try building *Capacity* faster by lowering the *time to adjust capacity*.

- ▶ Type the name *cust4* into the **Runname** editing box.
- ▶ Click on the button **Set Up a Simulation**.
- ▶ In the Capacity view, click on the highlighted Constant *time to adjust capacity* and type in the number 4 then press Enter.
- ▶ Click on the **Simulate** button.
- ▶ Select the variable *new customers* as the Workbench Variable and click on the **Graph** tool.



Now we see that *new customers* has moved closer to the ideal of the first model *cust1.mdl*. The *Capacity* constraint has been pushed back, though not eliminated.

9 Customizing Output

Output from Graph Analysis Tools

Graphs and Strip Graphs can display lines with attached numbers and symbols.

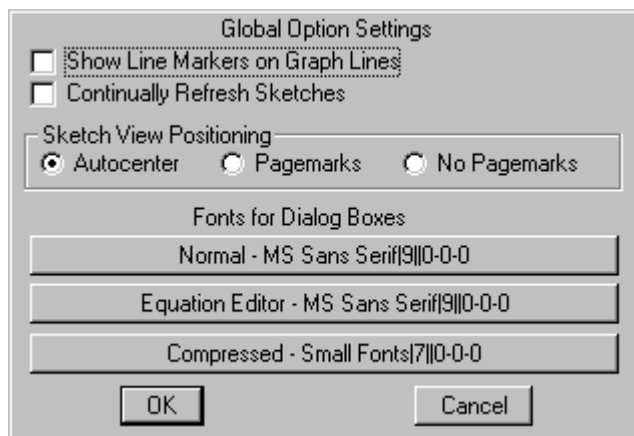
Graph tool.

EITHER

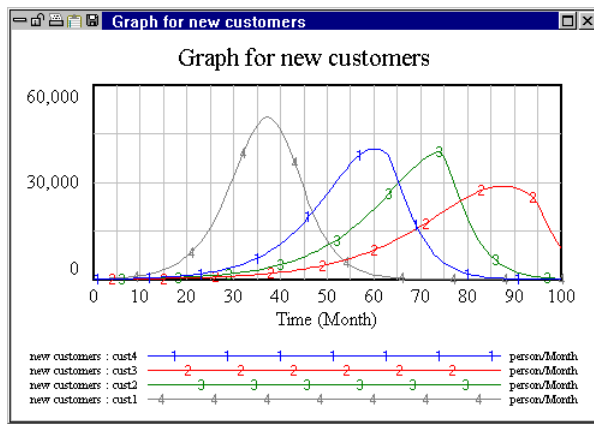
- Open the *cust3.mdl* model you created in Chapter 8

OR

- Open the model *cust3.mdl* in the directory *plemodel\chap08\complete*.
- Click on the **Control Panel** button then on the **Datasets** tab. Check to see that the runs *cust1* through *cust4* are all loaded. If they are not loaded, double click on each run in the **Available** list box to load them.
- Double click on the variable *new customers* appearing in the sketch to select it as the Workbench Variable.
- Select the menu item **Options>Options**. The Global Options dialog will open.



- Click on the **Show Line Markers on Graph Lines** checkbox so that it is checked. Click on **OK**.
- Click on the **Graph** tool.



Reset the Global Options dialog to defaults by:

- Select the menu **Options>Options....**. Click on the **Show Line Markers on Graph Lines** checkbox so that it is not checked. Click on **OK**.

NOTE Changing this setting on the Global Options dialog will change the appearance of existing graphs both on the screen and when they are printed.

Custom Graphs

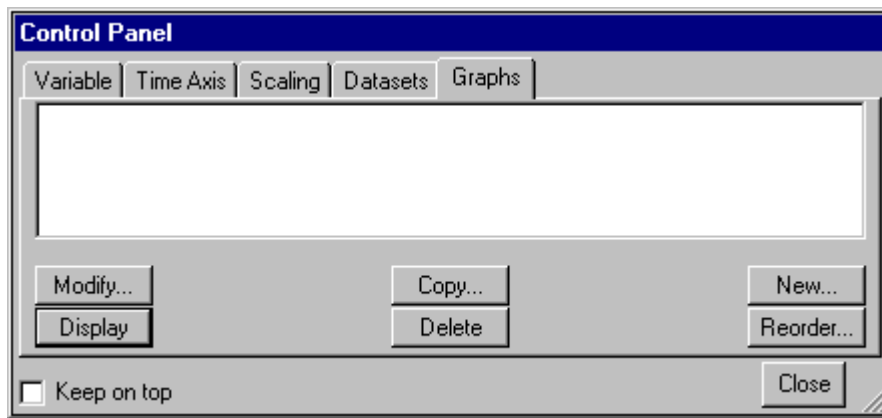
Custom Graphs are used to customize the content of a graph so as to show specifically selected variables, runs, and style, in one graph. Custom Graphs are created from the Graph Control in the Control Panel.

If *cust3.mdl* is not already open:

- Open the model *cust3.mdl* in the directory *plemodel\chap08\complete*.

To make a Custom Graph:

- Click on the **Control Panel** button on the Toolbar. Click on the tab **Graphs** in the Control Panel.



- ▶ Click the button **New....** The Custom Graph Editor opens with the cursor positioned at the graph **Title** editing box.
- ▶ Type in a name for the graph (e.g., Customer Diffusion)
- ▶ Using the mouse, move to the **Variable** boxes on the left side of the graph editor and click on the top button labeled **Sel.** A Variable Selection dialog box appears. Move the scrollbar down the list and double click on *Customers* (or type the first letters in Customers until it is highlighted, then press Enter).
- ▶ Click on the second button down labeled **Sel.** Double click on the variable *Customers*.
- ▶ Click on the third button down labeled **Sel.** Double click on the variable *Capacity*.
- ▶ Click on the fourth button down labeled **Sel.** Double click on the variable *Capacity*.
- ▶ Click the **Scale** checkbox that lies left and between the first two variables.
- ▶ Click the **Scale** checkbox that lies left and between the third and fourth variable.
- ▶ Click on the **Dataset** editing box just right of the first variable. Type in the run name *cust2*.
- ▶ Click on the **Dataset** editing box just right of the second variable. Type in the run name *cust4*.
- ▶ Click on the **Dataset** editing box just right of the third variable. Type in the run name *cust2*.
- ▶ Click on the **Dataset** editing box just right of the fourth variable. Type in the run name *cust4*.
- ▶ Click on the **LineW** (Line Width) editing box just right of the first variable, type 2.
- ▶ Click on the **LineW** (Line Width) editing box just right of the second variable, type 2.

The Custom Graph Editor should look like this:

Graph Name Hide: ☐ Title ☐ X Label ☐ Legend

Title

X-Axis Sel X Label

X-min X-max X-divisions Y-divisions

Stamp Comment

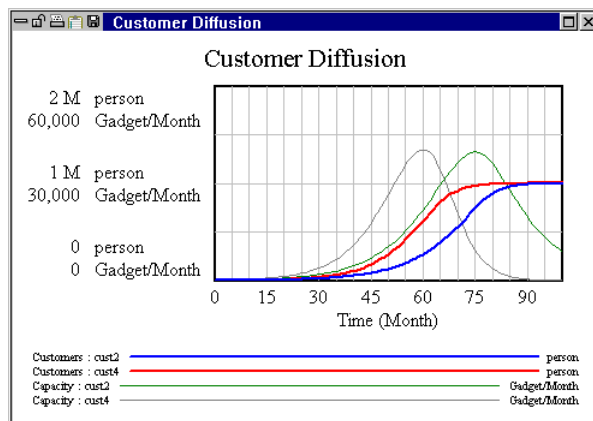
☐ Norm ☐ Cum ☐ Stack ☐ Dots ☐ Lbl-Intervals Width Height

Scale	Variable		Dataset	Label	LineW	Units	Y-min	Y-max
<input checked="" type="checkbox"/>	Customers	Sel	cust2		2			
<input type="checkbox"/>	Customers	Sel	cust4		2			
<input checked="" type="checkbox"/>	Capacity	Sel	cust2					
<input type="checkbox"/>	Capacity	Sel	cust4					
<input type="checkbox"/>		Sel						
<input type="checkbox"/>		Sel						

☐ As WIP Graph (maxpoints) Copy to... Test output ☐ Soft Bounds

OK Cancel

- Click on **OK**. The Custom Graph Editor closes and the Graph Control is left open.
- Click on the name of your graph in the Graph Control, then click on the button **Display**.



The graph is displayed, displaying the variables. You can easily see how the behavior of one variable relates to another. Experiment with different options in the Custom Graph Editor by clicking on the button **Modify** in the Graph Control, and changing things. If you do not include a dataset next to a variable name in the graph dialog, the **Display** button will show the first loaded run (the run at the top of the **Loaded** list in the Datasets Control).

What Are Games?

Games are a way of actively engaging in a model. Games are examples of the "flight simulator" approach, where the user participates in decisions that affect the simulation outcome for each step in time. A Vensim simulation model can be run as a game by stepping slowly through time and making changes to gaming variables along the way. In contrast, a normal simulation model runs through the complete time span based on the initial setup of the model.

The functionality discussed in this chapter is only available in Vensim PLE Plus. While Vensim PLE will open and simulate the model developed here, it does not have any functionality for running games.

The Real Estate Game (*houses.mdl*)

EITHER

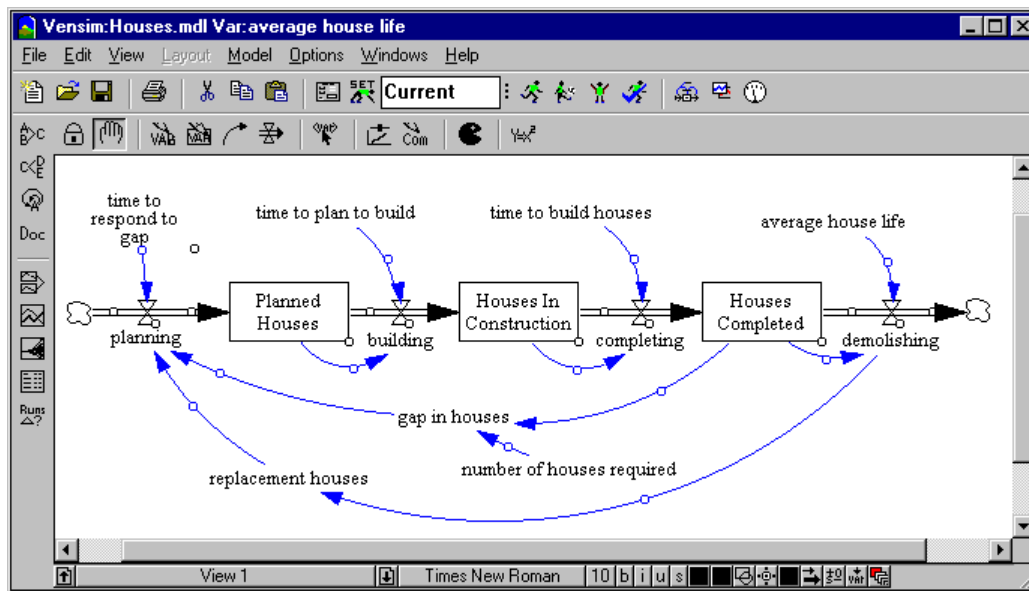
- Click the **Open Model** button and open the model *houses.mdl* in the directory *plemodel\chap10*.

OR

- Build the model as shown in the diagram and equation listing below and save it in the directory *plemodel\chap10* with a different name (e.g., *myhouses.mdl*). Time Bounds are INITIAL TIME = 0, FINAL TIME = 100, TIME STEP = 0.5, Units for Time: Month.

Model Structure

Below is a model of construction in the real estate industry. A long delay exists between the need to close the gap in houses and the completing of construction of those houses. This model features a negative feedback loop with several delays.



houses.mdl Equations

average house life = 1200

Units: Month

building = *Planned Houses* / *time to plan to build*

Units: house/Month

completing = *Houses In Construction* / *time to build houses*

Units: house/Month

demolishing = *Houses Completed* / *average house life*

Units: house/Month

gap in houses = *number of houses required* - *Houses Completed*

Units: house

Houses Completed = INTEG(*completing* - *demolishing* , 5000)

Units: house

Houses In Construction = INTEG(*building* - *completing* ,
building * *time to build houses*)

Units: house

Planned Houses = INTEG(*planning* - *building* ,
planning * *time to plan to build*)

Units: house

planning = MAX (0 , *replacement houses* + (*gap in houses* /
time to respond to gap))

Units: house/Month

```

replacement houses = demolishing
Units: house/Month

time to build houses = 6
Units: Month

time to plan to build = 3
Units: Month

time to respond to gap = 8
Units: Month

```

Built-in Functions

We are going to start this model in equilibrium. We will set *number of houses required* to 5000, which is the initial value for *houses completed*. Because of this, *gap in houses*, will be 0 so that *planning* will be equal to *replacement houses* which is equal to *demolishing*. Because of the way we initialized the other levels (for example *Planned Houses* is initialized equal to *planning * time to plan to build*) each of these is already in equilibrium. We should therefore have a model that will simulate without change.

While it is important to check to be sure that our understanding is correct and the model will simulation without anything changing, we also want to see more model behavior. Thus, instead of using just 5000 for *number of houses required* we want to hold it at 5000 for a time (say ten months) and then increase it (say to 5050). To do this we use the equation:

```

number of houses required = 5000 + STEP ( 50, 10 )
Units: house

```

The STEP function takes two arguments, **height** and **start time**, which are enclosed in parentheses. It takes on a value of 0 until **start time** is reached and from then on takes on value **height**. This function is a particularly good input for a model because it is a simple input change that generates a broad behavior response. Other functions that are useful for "exciting" or "disturbing" a model in this way include PULSE and RAMP.

To add the above equation open the Equation Editor on *number of houses*.

- ▶ Type in the number 5000, then the plus sign.
- ▶ Click on the **Functions** tab then scroll down until you see the STEP function appear in the list. Click once on the STEP function.
- ▶ The argument {height} should be highlighted, just type over it with the value 50.
- ▶ Double click on the {stime} argument and type in 10.
- ▶ Enter the units *house* and click **OK**.
- ▶ Use the command Model>Check Model (or Ctrl+T) to check your model.

If there are any errors compare your equations to those shown here and correct any mismatches.

10: Vensim PLE User's Guide

- Create the Custom Graph definition shown below. Be sure to check the **As WIP Graph** checkbox (this checkbox is only available in Vensim PLE Plus). Chapter 9 describes how to create Custom Graphs.

The screenshot shows the 'Custom Graph' dialog box in Vensim. The 'Graph Name' is 'REAL_ESTATE_GAME'. The 'Title' is 'Real Estate Game'. The 'X-Axis' is empty, and the 'X Label' is empty. The 'X-min' is 0, 'X-max' is 100, 'X-divisions' is empty, and 'Y-divisions' is empty. The 'Stamp' is empty, and the 'Comment' is empty. The 'Scale' is 'Norm', 'Cum' is unselected, and 'Stack' is unselected. The 'Dots' checkbox is unselected, and 'LbIntervals' is unselected. The 'Width' is empty, and 'Height' is empty. The 'Table' has the following data:

Scale	Variable	Dataset	Label	LineW	Units	Y-min	Y-max
<input type="checkbox"/>	Houses Complete	Sel		2		5000	5150
<input type="checkbox"/>	planning	Sel		2		0	20
<input type="checkbox"/>	gap in houses	Sel		3		-100	100
<input type="checkbox"/>		Sel					
<input type="checkbox"/>		Sel					
<input type="checkbox"/>		Sel					

The 'As WIP Graph (maxpoints)' checkbox is checked, and the 'maxpoints' value is 200. The 'Copy to...' button is highlighted. The 'Test output' button is also visible. The 'Soft Bounds' checkbox is unselected. The 'OK' and 'Cancel' buttons are at the bottom.

Adding Game Variables

The goal in playing this game is to meet the demand for houses (a zero *gap in houses*); you do this by setting and changing the variable *planning* which introduces new houses into the planning and construction process. Right now *planning* is determined by a formula. This formula allows you to simulate the model, but does not provide a mechanism for you to intervene and change the value of *planning* during a simulation. You need to change *planning* to a Game variable. To do this:

- Select the **Equations** tool.
- Click on the variable *planning*.

We see the equation:

```
planning = MAX( 0, replacement houses + (gap in houses /  
time to respond to gap))
```

Units: house/Month

The equation is formulated so planning can never go negative; you can plan to build some houses (positive) or plan to build no houses (zero). To make this a Game variable, we change the variable type in the lower drop-down variable type box:

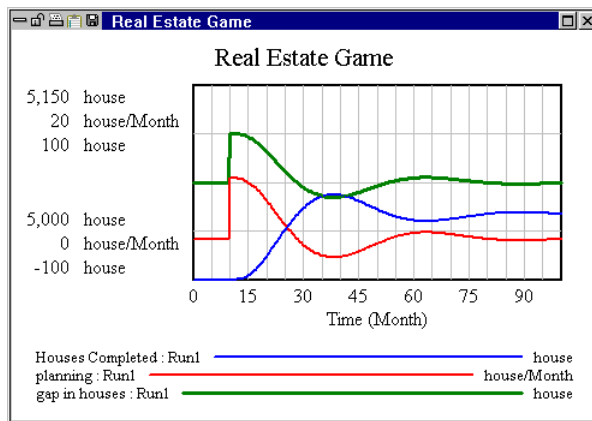
- Click on the drop-down arrow on the lower variable **Type** box (currently reads **Normal**) and choose **Gaming** from the list. Click the **OK** button.
- Save the model.

You can convert any Auxiliary, Rate or Constant into a Game variable in this manner. During a simulation a Game variable does just what it would do if it were an Auxiliary, Rate or Constant. However, during a game, you can set the value of a Game variable at each time as the game progresses.

Simulating the Model

Before playing the game, let us look at how the model behaves when simulated.

- Double click on the **Runname** editing box, type in *run1* (or any name), then click the **Simulate** button.

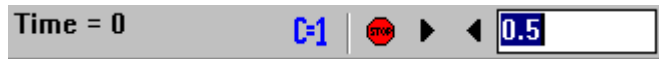


A Work-In-Progress (WIP) custom graph is generated showing behavior for three key variables in the model: *Houses Completed*, *planning*, and *gap in houses* (the thickest line). Note the oscillation: overshoot then undershoot of the goal. The model is trying to drive the *gap in houses* to zero. The step in the model is coming from *number of houses required*. Let us see if we can do any better by planning houses ourselves to try and keep a zero *gap in houses*.

Playing the Game

- ▶ Double click on the **Runname** editing box, type in *game1* (or any name), then click the **Game** button.

A WIP (Work-In-Progress) Custom Graph is generated and the Toolbar changes to the Game Toolbar:



The Game **Time** is displayed on the left. The **Change Gaming Variables** button (C=1) provides one way to change the value of Game variables during the game. The **Stop** button stops the game. **Move Forward** and **Move Backward** buttons move the game by the amount shown in the **Amount to Move** editing box (currently displaying 0.5). The three standard window class buttons are on the right.


Moving Forward in a Game

- ▶ Move the WIP graph to the right of your screen.


You should see that the variable *planning* is highlighted yellow with blue text. This provides the second way to change the value of Game variables during the game.

- ▶ Click on the variable *planning*, you will see its initial value (4.166), press Enter to leave without changing the value.

Note that the WIP graph disappeared behind the Build window. If we change the Game variable from the toolbar using the button **Change Gaming Variables**, the WIP graph will stay on top. Another way to keep the two windows visible is to reduce the size so both fit on your screen.

- ▶ Click on the reduce button  for the Build window (upper right corner, but below the reduce button for the Vensim application).

The Build window containing the sketch will shrink to a smaller size.

- ▶ Resize and position the Build window and the graph window so you can see both the model (or at least the variable *planning*) and the WIP graph.
- ▶ Double click on the **Amount to Move** editing box on the Game Toolbar and type in **5**.
- ▶ Click on the **Move Forward** button .

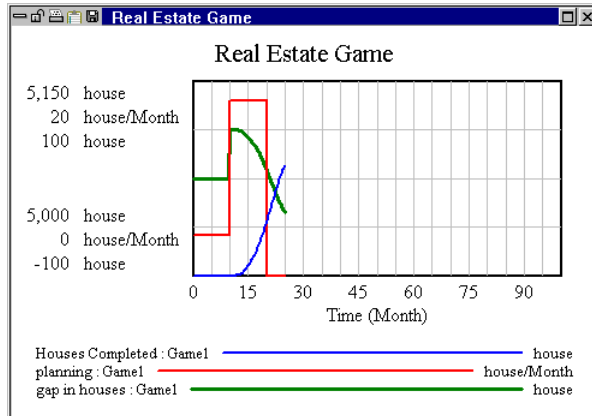
The WIP graph will start to draw. Behavior is in equilibrium; we do not need to change any parameters because the *gap in houses* is currently zero (exactly on the middle gridline).

- ▶ Click on **Move Forward** once more, you will see the step upwards in the *gap in houses*.
- ▶ Click on *planning* on the sketch, type a value of 18, then press Enter.

- Click on **Move Forward** two more times.

See how *gap in houses* reduces, while *Houses Completed* takes an upturn. We have almost closed the gap to zero. We had better stop building so many houses.

- Click on *planning* on the sketch, type in 0, then press Enter.
- Click on **Move Forward**.



Wow! We have overshot the mark; our goal (*gap in houses*) is now negative (below the middle gridline). Since we cannot plan negative houses, we had better plan zero houses for a while.


- Click on **Move Forward** until the *gap in houses* is positive (just above zero at about Time = 50).

Now we should start building again so that we don't get a positive *gap in houses* (where more houses are required). We can anticipate this somewhat by building a little before *gap in houses* is positive.

Backing Up in a Game

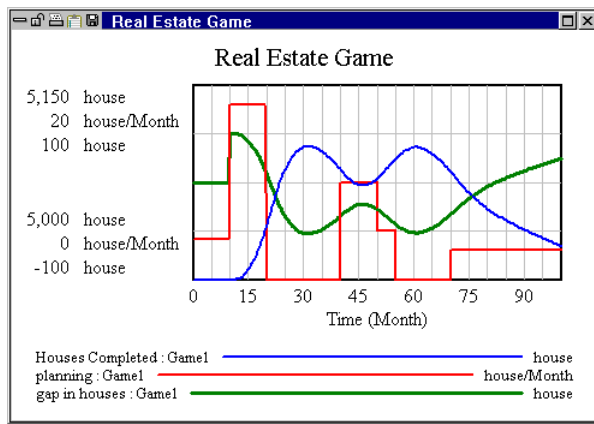
- Click on the **Move Backward** button  twice (until *gap in houses* is negative).

Of course, we can't back up the real world. But in order to try out different options when the game has progressed too far in some direction, we can back up in the game.

- Click on *planning* on the sketch, type in a larger number (for example, 10), then press Enter.
- Click on **Move Forward**.
- Continue playing the game, trying to keep the *gap in houses* at or near zero until the final time of 100 is reached.
- Click the **Stop** button .

Your WIP graph will end up looking something like this:

10: Vensim PLE User's Guide



Your game results are probably not much better (and maybe a lot worse) than the original simulation. In the graph above, *gap in houses* (which we tried to keep at zero) fluctuates wildly in response to our *planning* decisions.

11 Input Output Controls

You can customize the way your sketches work with simulation models by adding Input and Output Controls to model views. These can be added alongside model structure, or in separate views from model structure. Using Input Output Controls you can build your own "control room" for managing model inputs and viewing simulation results.

In this chapter you will build a control room for an existing model. Input Output Controls are not part of model structure and, therefore, do not influence model behavior. They can easily be added to a finished model to make it easier for another person to use the model. The Controls are supported in the Vensim Model Reader and provide a simple mechanism to make your models easier to consume. Input Output Controls also adapt to changes in model structure quite well. If you rename a variable the corresponding control will automatically be updated. If you delete a variable or change its type any associated controls will no longer work, but will not stop you from working on the model. Rather they will simply appear blank or inactive.

Input Output Controls are only available in Vensim PLE Plus. You can use Vensim PLE to open and simulate models containing Input Output controls. These controls will, however, remain inactive. In addition Vensim PLE does not support Navigation Links or binary format saves.

Word of Mouth Sales

For this example we will use a word of mouth sales model.

EITHER

- Open the model *wom1.mdl* contained in the directory *plemodel\chap11*.

OR

- Build the model as shown in the diagram and equation listing below and save it in the directory *chap11* with a name such as *mywom.mdl*. Time Bounds are INITIAL TIME = 0, FINAL TIME = 5, TIME STEP = 0.0625, Units for Time: Year.

wom1.mdl Equations

```
advertising effectiveness = 0.1
Units: Widget/$

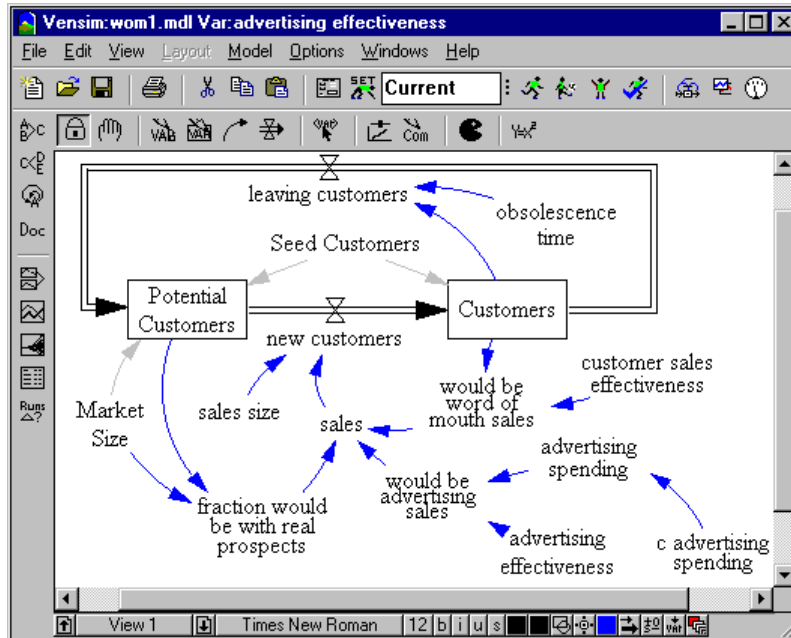
advertising spending = GAME( c advertising spending )
Units: $/Year

c advertising spending = 2e+007
Units: $/Year

customer sales effectiveness = 3
Units: Widget/(Year*Person)
```

11: Vensim PLE User's Guide

Customers = INTEG(new customers - leaving customers, Seed Customers)
Units: Person



fraction would be with real prospects =
Potential Customers / Market Size

Units: Dmnl

leaving customers = Customers / obsolescence time
Units: Person/Year

Market Size = 1e+008
Units: Person

new customers = sales / sales size
Units: Person/Year

obsolescence time = 2
Units: Year

Potential Customers = INTEG(- new customers + leaving customers ,
Market Size - Seed Customers)
Units: Person

*sales = (would be word of mouth sales + would be advertising sales) **
fraction would be with real prospects
Units: Widget/Year

sales size = 1
Units: Widget/Person

Seed Customers = 10000

Units: Person

*would be advertising sales = advertising spending *
advertising effectiveness*

Units: Widget/Year

*would be word of mouth sales = Customers * customer sales effectiveness*


Units: Widget/Year

Output Controls

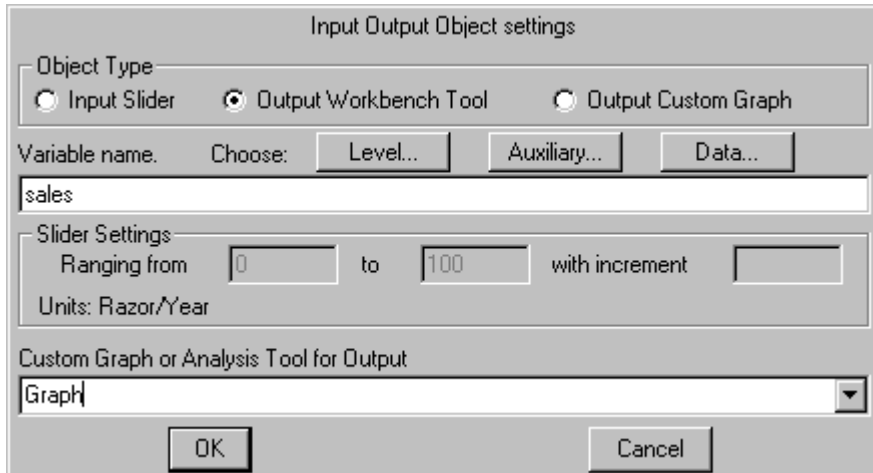
Begin by creating a new view to which we can add Output Controls.

- Click on the Status Bar View button (this is at the bottom of the screen and labeled **View 1** for this model) and select ****New****.

A new and empty View will open.

- Select the **Input Output** tool  by clicking on it, or by pressing the 8 (eight) key.
- Move the mouse to the right hand side of the sketch and click. This will open the Input Output Object Settings dialog.
- Select type **Output Workbench Tool** then click on the **Auxiliary** button and Select *sales* from the list (clicking OK to close the Variable Selection dialog).
- Click on the dropdown and select the tool **Graph** from the dropdown list.

Your dialog should look like:

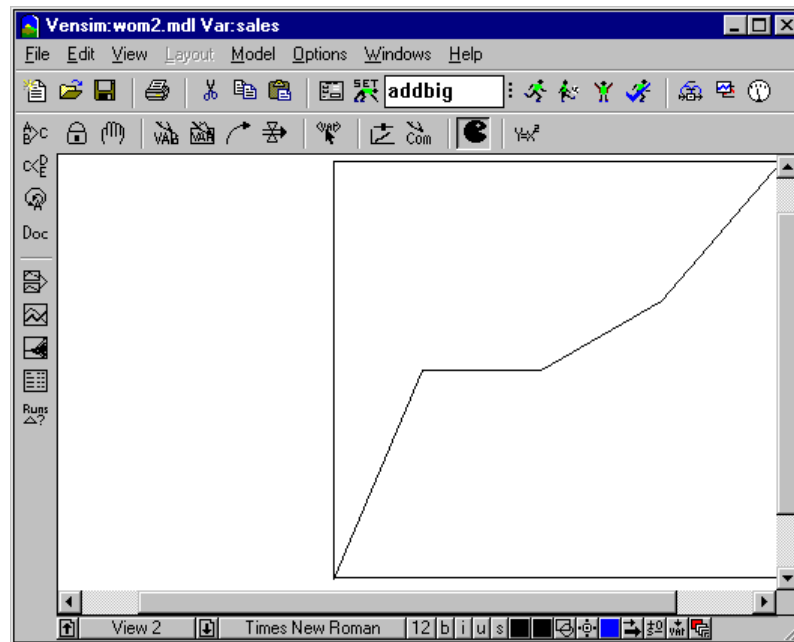



The dialog box is titled "Input Output Object settings". It contains three radio buttons under "Object Type": "Input Slider" (unselected), "Output Workbench Tool" (selected), and "Output Custom Graph" (unselected). Below these are three buttons: "Level...", "Auxiliary...", and "Data...". The "Variable name" field contains the text "sales". Under "Slider Settings", there are fields for "Ranging from" (0), "to" (100), and "with increment" (empty). The "Units" field contains "Razor/Year". At the bottom, there is a dropdown menu labeled "Custom Graph or Analysis Tool for Output" with "Graph" selected. At the very bottom are "OK" and "Cancel" buttons.

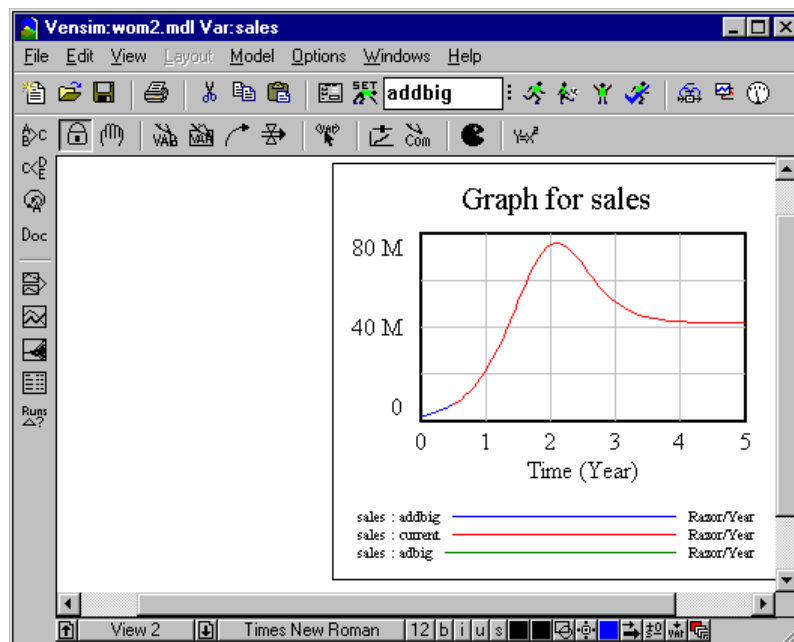
- Click on **OK**.

11: Vensim PLE User's Guide

You will see a fairly large Rectangle with the sizing handle visible. Resize this to fill roughly the right hand side of the sketch. Your model should look like this:




- Click on the **Simulate** button  on the Toolbar (or type Ctrl+R). The model will simulate and the graph will fill in.



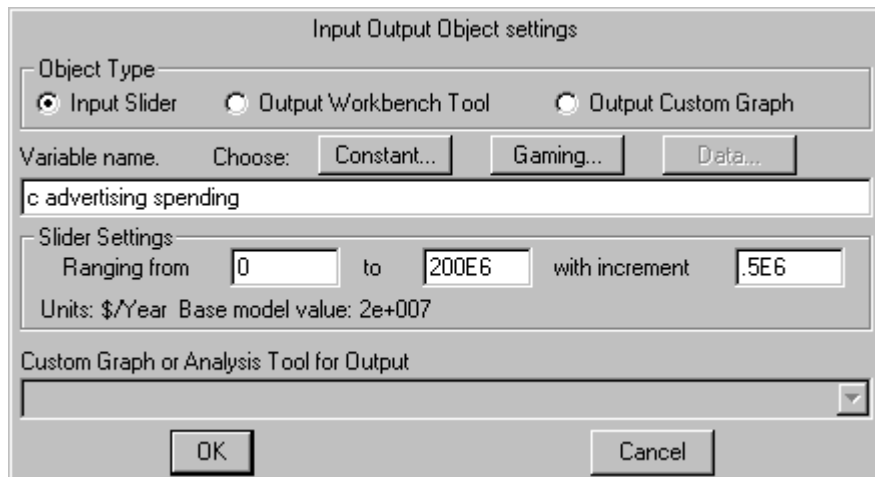
Input Controls

We will continue working in the same view.

Slider for Advertising

- ▶ Select the **Input Output** tool  and click on the upper left portion of the sketch. The Input Output Object Settings dialog will open.
- ▶ Select type **Input Slider** (probably already selected by default).
- ▶ Click on the **Constant...** button and select *c advertising spending* from the dialog.
- ▶ Enter the numbers 0, 200 E6, and 0.5E6 in the **Ranging from**, **to**, and **with increment** fields.

Your dialog should look like:




- ▶ Click on **OK**.
- A Slider will be drawn on the screen.
- ▶ Adjust its size so that it more or less takes up the space to the graph.
 - ▶ Select the **Comment** tool. Click above the Slider.
 - ▶ Type in the comment "Advertising Spending (\$/Year)," select Shape **None** and click on OK. This is a label for the Slider you just created.


Slider for Product Life

Now repeat this entire process creating a Slider for *obsolescence time* with a range from 0.5 to 10 and increment of 0.1. Place the label "Product Lifetime (years)" above this slider.

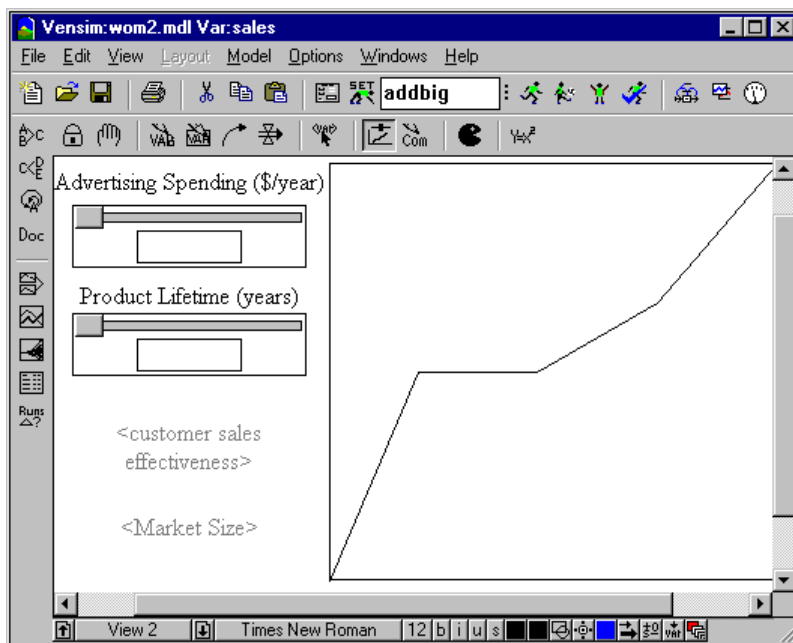
Other Changeable Constants

- ▶ Select the **Shadow Variable** tool  and add in the model Constants *customer sales effectiveness* and *Market Size*.
- ▶ Highlight the new Shadow Variables and set the color on them to black from the Status Bar color button, or Right-Click on them in turn and set their text color to black.

Lining Things Up


- ▶ To make the things you have added line up nicely select the **Size/Move** tool  then drag over all the elements on the left hand side to highlight them.
- ▶ Shift-Click on the topmost comment (Advertising Spending (\$/year)) once to unhighlight it, and again to highlight it.
- ▶ From the **Layout** Menu select the item **Center on LastSel**.
- ▶ Position the whole selection by dragging the middle of the selection box until it looks good.

You should end up with.



Now is a good time to save your work under a new name. The changes that have been made to *wom1.mdl* are also saved in the model *wom2.mdl* contained in the directory *plemodel\chap11\Complete* if you want to compare them with your own changes.

Simulating the Model

- ▶ Click on the **Set Up a Simulation** button in the Toolbar  (or type Ctrl+E). Type in a new run name.


The Graph will fill in, the Sliders will activate, and the two shadow variables will highlight.

- ▶ Drag a slider and then click on the **Simulate** button in the Toolbar  or type (Ctrl+R).

The model will simulate and the results for the new run will be displayed in the graph. Repeat this making other changes. You can change the run name or keep it the same. Each time you simulate the results will be displayed in the graph on the right.

Gaming Control

The word of mouth model has one gaming variable *Advertising Effectiveness*. We can use the same basic screen layout that we did for the simulation control setup.


- ▶ Select the **Move/Resize** tool .
- ▶ Highlight the Output Object on the right, the Comment "Advertising Spending (\$/Year)" and the Slider below this.

You can do this by Clicking on one, then Shift-Clicking on the other two.

- ▶ Select **Edit>Copy** (or Ctrl+C).

NOTE It is easiest to copy Input Output Objects with the Move/Resize tool active. When you activate the Lock tool these objects no longer get selected on Click or Shift-Click, though they can be selected by dragging around them.

Pasting to a New View

- ▶ Start a new view by clicking on ****New**** in the Status Bar View button at the bottom.
- ▶ Use the Edit command Paste (or Ctrl+V).
- ▶ Click OK on the paste dialog. There isn't really any structure to paste so **Replicate** and **Paste Picture** have the same effect.
- ▶ Select the **Input Output** tool  and then click on the Slider.

The Input Output Control dialog will open.

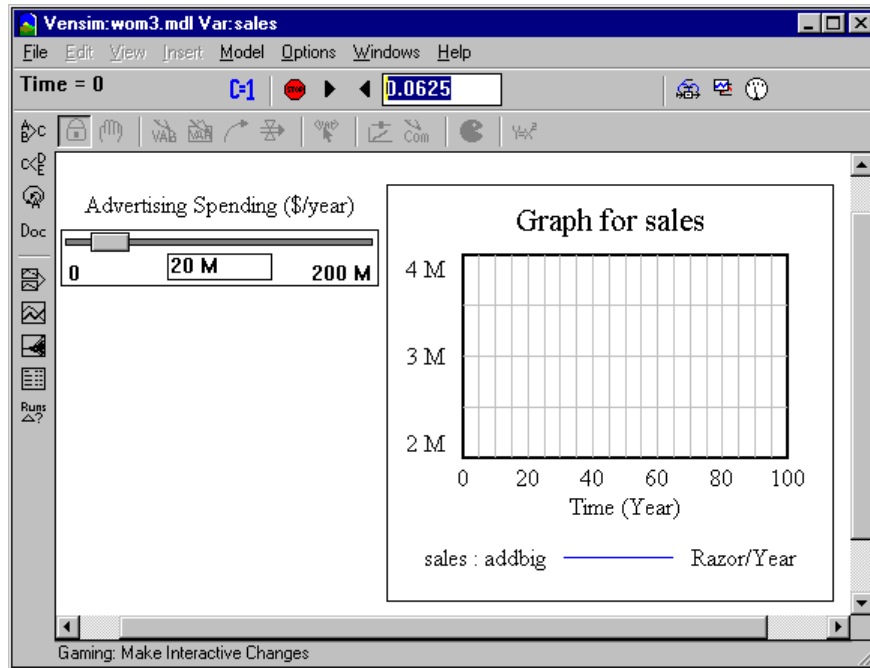
- ▶ Click on the **Gaming...** button and click select *advertising spending* (the only thing in the list). Click on **OK**.

Note that you do not need to change the label, since it is still advertising spending that is being controlled.


Running a Game


- Select the Control Panel **Datasets** tab and unload all datasets.
- Click on the **Game** button  to start a game.

The graph will fill (perhaps with a full time scale but possibly with only a 0-100 scale if you are overwriting the only loaded run). The scale should adjust as you move forward in time. The Slider will become active and the Toolbar will change to reflect the gaming state:



The Gaming Interval appears highlighted in an editing box in the Toolbar. By default the Gaming Interval is set to the TIME STEP which is 0.0625.

- Type in 0.25.
- Make changes using the Slider and click on the advance button  to move forward in time.

The graph will update. You can also move back in time using the backup button .

Now is a good time to save your model. The changes that have been made here are also saved in the model *wom3.mdl* contained in the directory *tutorial\chap11\Complete* if you want to compare them with your own changes.

Publishing the Model

The model as it now stands has a simple interface for changing constants and running simulations and it also has a simple interface for running games. You can help people who have no knowledge of the model by adding navigation buttons for the Views, and also build in a Gaming Interval that is more appropriate than the default TIME STEP..

Game Interval

To set the default Game Interval to be a more reasonable number simply add the Constant *Game Interval* to the model, mark it as a Supplementary variable and set its value equal to 0.5 (Years). When a game is started, the Constant 0.5 will be read in as the interval for Game steps.

Commentary and Navigation Links

To help a newcomer to the model, it is appropriate to set up some instructions and guides to help them navigate around.

- ▶ Create a new View by selecting ****New**** on the Status Bar **View** button.

Renaming Views

- ▶ Select menu **View>Rename**. In the dialog that opens type in "Overview" and click on **OK**.

Your view will now be called "Overview." You should rename the other views in the same manner to "Structure," "Simulate," and "Game."

Reordering Views

- ▶ Select menu **View>Reorder**.
- ▶ Press the mouse button on "Overview" in the list of Views.
- ▶ Holding down the mouse button move the mouse up to near the top.

"Overview" should disappear and the shape of the pointer should change to a crosshair.

- ▶ Move the crosshair up so that it is centered near the top of the first name (Structure) and let go of the mouse button.


"Overview" should appear in the first position. If it does not just repeat the operation letting go of the mouse a little lower. If you move the center of the crosshair outside of the list "Overview" will just drop back to its old position.

- ▶ Click **OK**.

Adding Commentary

If you are not currently on the "Overview" View, move there by selecting "Overview" from the bottom View button or by using the Page Up and Page Down keys.

11: Vensim PLE User's Guide

- ▶ Select the **Comment** tool  and click on the top part of the screen.
- ▶ Type in some commentary you think will be helpful. See below to see what comments we added.

Navigation Links

A Navigation Link is just a comment with the **Navigate** field filled in. When you click on a Navigation link (with the **Lock** tool selected) the View is automatically changed to the View named in the **Navigate** field.

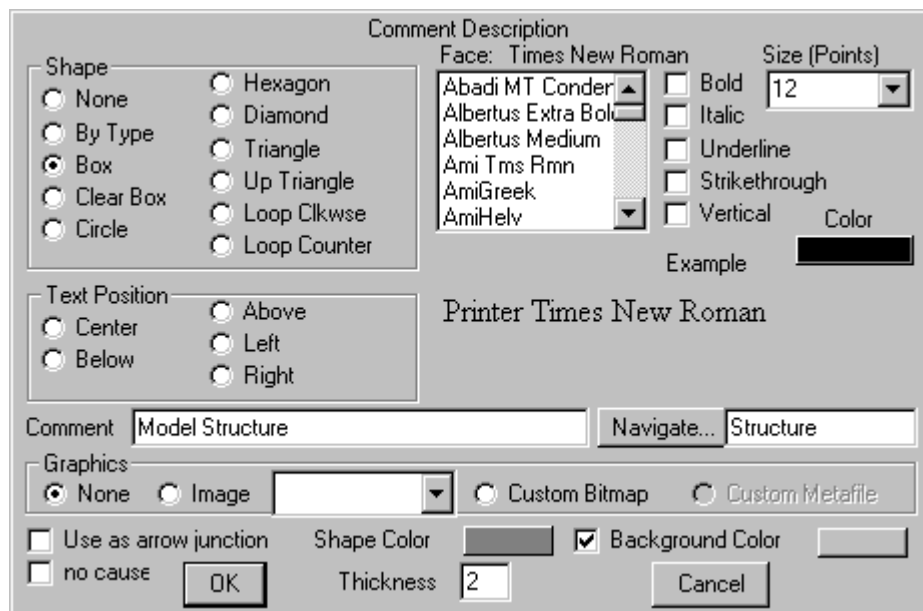
- ▶ With the **Comment** tool, click on the diagram below your existing comments.
- ▶ Type in "Model Structure" and select **Box** in the **Shape** field.
- ▶ Click on the **Navigate...** button

A list of views will appear in a new dialog.

- ▶ Select "Structure" from the list and click on **OK**.
- ▶ Click on the button to the right of **Shape Color** and click on a dark gray button.
- ▶ Click on the button to the right of **Background Color** and click on a light gray color.
- ▶ Fill in the field **Thickness** below the colors with two.

The use of a gray color and dark gray box with **Thickness** set to 2 makes the Comment look a little more like a button.

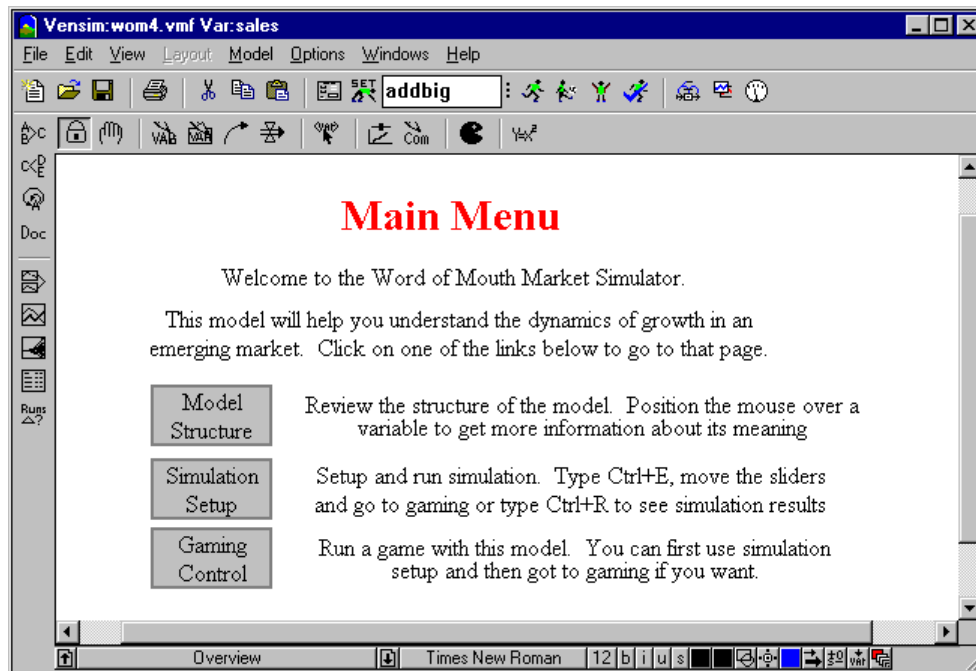
The comment description dialog should appear as:



The image shows the 'Comment Description' dialog box in Vensim. It has several sections: 'Shape' with radio buttons for None, By Type, Box (selected), Clear Box, Circle, Hexagon, Diamond, Triangle, Up Triangle, Loop Clkwise, and Loop Counter; 'Text Position' with radio buttons for Center, Below, Above, Left, and Right; 'Face' with a list box showing fonts like Abadi MT Conder, Albertus Extra Bold, etc.; 'Size (Points)' with a dropdown set to 12; checkboxes for Bold, Italic, Underline, Strikethrough, and Vertical; a 'Color' button; an 'Example' preview showing 'Printer Times New Roman'; a 'Comment' text field with 'Model Structure'; a 'Navigate...' button; a list box with 'Structure'; 'Graphics' options for None (selected), Image, Custom Bitmap, and Custom Metafile; checkboxes for 'Use as arrow junction' and 'no cause'; 'Shape Color' and 'Background Color' buttons; and a 'Thickness' text field set to 2. At the bottom are 'OK' and 'Cancel' buttons.

- Set up Navigation Links to each of the other views giving them the labels "Simulation Setup" and "Gaming Control."

You should have a diagram that looks like this:



- Add in Navigation links back to the Overview View from each of the other Views. In the Simulation View you also need to add a Navigation link to the Game View.

Now is a good time to save your model. The changes that have been made here are also saved in the model *wom4.vmf* contained in the directory *plemodel\chap11\Complete* if you want to compare them with your own changes. The *.vmf* file extension used instead of the normal *.mdl* extension and this is discussed below.

Test It Out

Select the Lock tool and try the Navigation links. Also repeat the process of setting up and running a simulation. Review the appearance of the "Game" view in simulation setup mode. Review the appearance of the "Simulate" view in Gaming mode. The behavior of your model with the Lock tool selected is essentially the same as its behavior will be in the Vensim Model Reader.

Binary Format Save

The Vensim model reader is a read only program that has no capacity to convert text format models into something that can simulate. Because of this it is necessary to save the models you develop in a special binary form before they can be used in the Vensim model reader. To do this just choose

11: Vensim PLE User's Guide

File>Save As... from the File menu, select the type **Binary Format Models** and then type in a name. You can also just type in the name with extension .vmf (as in *wom4.vmf*) and Vensim will determine the type of file you want to save as from the extension you have typed.

While you can open and work with binary format models their only value with Vensim PLE Plus is for the publishing of your models.

You have now created a model that you can send to anyone and they can use by downloading the free Vensim Model Reader.

The functionality covered in this chapter is only available in Vensim PLE Plus. Sensitivity Testing is a way of working with models and the models do not need to be modified to perform Sensitivity Testing. Therefore, setting up and performing Sensitivity Testing on a model will not prevent someone from using that model with Vensim PLE.

Monte Carlo Simulations

Sensitivity testing is the process of changing your assumptions about the value of Constants in the model and examining the resulting output for change in values. Manual sensitivity testing involves changing the value of a Constant (or several Constants at once) and simulating, then changing the value of the Constant again and simulating again, and repeating this action many times to get a spread of output values.

Monte Carlo simulations, also known as multivariate sensitivity simulations (MVSS), makes this procedure automatic. Hundreds or even thousands of simulations can be performed, with Constants sampled over a range of values, and output stored for later analysis. Latin Hypercube sampling is a specialized form of sensitivity testing that allows faster sensitivity testing on very large models, or when computers have slow simulation speeds.

Market Growth Model (*sales.mdl*)

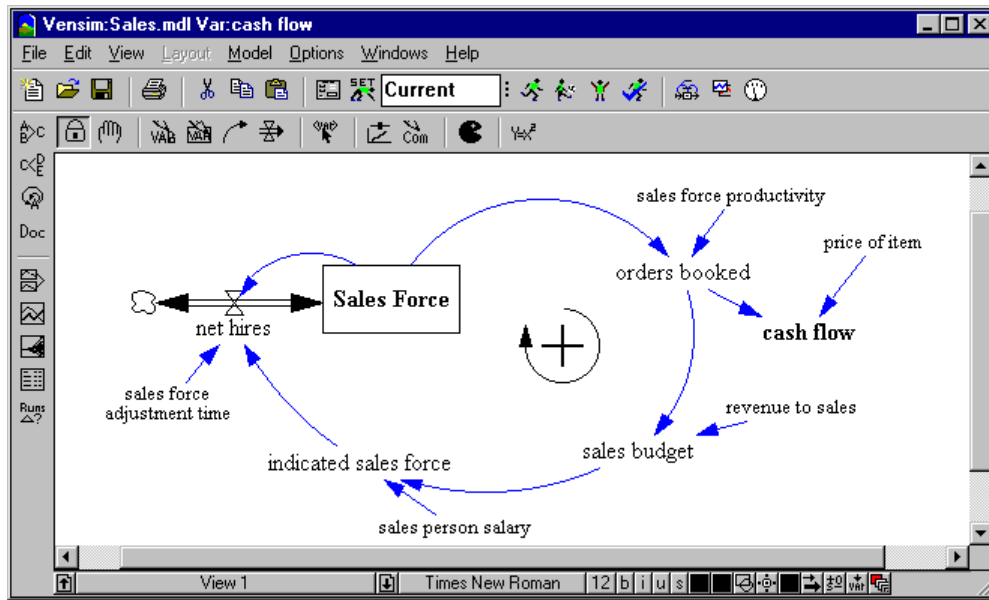
The sales model shown below contains a major positive feedback loop, where more sales people can generate more sales, thereby increasing revenue and allowing more sales people to be hired. A minor negative feedback loop allows adjustment of the sales force over a period of time (to hire or fire).

EITHER

- Open the model *sales.mdl* contained in the directory *plemodel\chap12*.

OR

- Build the model as shown in the diagram and equation listing below and save it in the directory *plemodel\chap12* with a different name (e.g., *sales1.mdl*). Time Bounds are INITIAL TIME = 0, FINAL TIME = 60, TIME STEP = 0.25, Units for Time: Month.



Sales.mdl Equations

```

cash flow=
    orders booked * price of item
Units: dollars/Month

indicated sales force=
    sales budget / sales person salary
Units: person

net hires=
    (indicated sales force - Sales Force)/sales force adjustment time
Units: person/Month

orders booked=
    Sales Force * sales force productivity
Units: unit/Month

price of item=
    100
Units: dollars/unit

revenue to sales=
    10
Units: dollars/unit
    
```

```

sales budget=
    orders booked * revenue to sales
Units: dollars/Month
Sales Force= INTEG (
    net hires,
    50)
Units: person
sales force adjustment time=
    6
Units: Month
sales force productivity=
    210
Units: unit/(person*Month)
sales person salary=
    2000
Units: dollars/(person*Month)

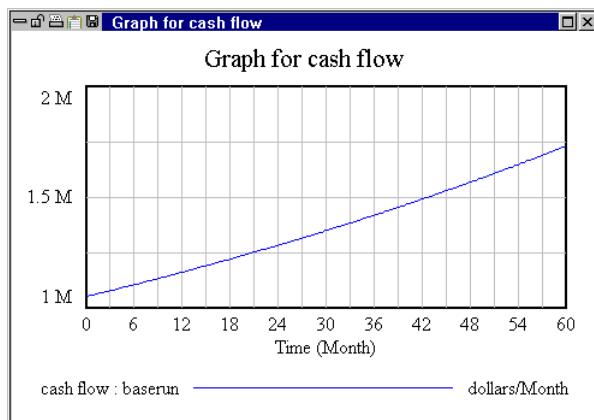
```

Base Simulation

First we will simulate the model to see the behavior with the normal values for the model's Constants.

- Double click on the **Runname** editing box, type in *baserun*.
- Click the **Simulate** button.
- Double click *Sales Force* then click on the **Graph** tool. Double click *cash flow* then click on the **Graph** tool.

We find that both *Sales Force* and *cash flow* are slowly growing.



- Select the Control Panel **Datasets** tab and double click on *baserun* to unload it.


Uncertainty in Multiple Parameters

This model contains five constants that we can vary to examine their effect on simulation output. We will assume that we know the exact values for two constants: *price of item* and *revenue to sales* (because these are policy decisions that managers can readily set). The uncertain parameters are *sales force productivity*, *sales force adjustment time*, and *sales person salary*. We will select these parameters and assign them maximum and minimum values and a random distribution over which to vary them to see their impact on model behavior. Note that we could select only one parameter if we wanted to see how sensitive model behavior is to one parameter.

NOTE Parameter is a synonym for Constant. In the Vensim documentation parameter is frequently used to refer to model Constants that we select for variation during sensitivity analysis and optimization.

Sensitivity Control Parameters

Vensim sets up sensitivity simulations from the Sensitivity button on the Toolbar which activates the Sensitivity Wizard.

- ▶ Double click on the **Runname** editing box and type the name *sensitivity* for the dataset (currently *baserun*).
- ▶ Click the **Sensitivity** button  on the Toolbar.

The Sensitivity Wizard opens at the Sensitivity Control

Sensitivity Simulation Setup

Sensitivity Control. Edit the filename to save changes to a different control file

Filename:

Number of Simulations:

Initial Noise Seed:

☐ Univariate (change one at a time)
☒ Multivariate (change all together)
☐ Latin Hypercube

Currently active parameters

Distribution:

Parameter	Minimum Value	Maximum Value
--	<input type="text"/>	<input type="text"/>

- Make sure that the radio button for **Multivariate** is selected and that the **Number of Simulations** is set to 200.

Monte Carlo multivariate sensitivity works by sampling a set of numbers from within bounded domains. To perform one multivariate test, the distribution for each parameter specified is sampled, and the resulting values used in a simulation. When the **Number of Simulations** is set at 200, this process will be repeated 200 times.

- Click on the **Parameter** button, a Control Parameter dialog box will open showing all the parameters (Constants) in the model that can be selected for Monte Carlo sampling. Click on *sales force productivity* and click **OK**.

Random Uniform Distribution

A distribution for sampling needs to be chosen. The simplest distribution is the Random Uniform Distribution, in which any number between the minimum and maximum values is equally likely to occur. The Random Uniform Distribution is suitable for most sensitivity testing and is selected by default.

Another commonly-used distribution is the Normal Distribution (or Bell Curve) in which values near the mean are more likely to occur than values far from the mean. Vensim provides a variety of different distributions to choose from. The most commonly used distributions are the Uniform, Normal and Triangular distributions.

12: Vensim PLE User's Guide

Minimum and maximum values are chosen to bound each parameter. Note the actual model value of 210 showing below the **Parameter** button.

- ▶ Click on the box labeled **Minimum Value** and type in 200. Click on the box labeled **Maximum Value** and type in 220.

The minimum value of 200 represents the lowest productivity we think the sales force can achieve, the maximum value of 220 represents the highest productivity we think they can achieve.

- ▶ Click on the **Add Editing** button.
- ▶ Click on the **Parameter** button and click on *sales force adjustment time* and click **OK**.
- ▶ Click on the box labeled **Minimum Value** and type in 3. Click on the box labeled **Maximum Value** and type in 12. These figures are asymmetrical around the model's value of 6; we think the value might be a little lower or a lot higher. Click on the button **Add Editing**.
- ▶ Click on the **Parameter** button and click on *sales person salary* and click **OK**.

Random Normal Distribution

Let us choose a different distribution from the default distribution. RANDOM NORMAL samples values according to a Normal Distribution¹, and requires maximum and minimum bounds and a mean and standard deviation to be specified.

- ▶ Click on the dropdown arrow of the box **distributed**. Select RANDOM NORMAL from the list (you will need to use the scrollbar to scroll the list up).

Note that some editing boxes have been added to the bottom of the Sensitivity Control.

- ▶ Click on the box **Minimum Value** and type in 1800. Click on the box **Maximum Value** and type in 2200. Click on the box **Mean** and type in 2000. Click on the box **Standard Deviation** and type in 100. Click on the button **Add Editing**.

The Sensitivity Control should look the same as below.

¹Technically this is called a Truncated Normal Distribution. The Normal Distribution is unbounded and very large negative and positive values can occur. You can truncate all distributions so that no extreme values are used.

Sensitivity Simulation Setup

Sensitivity Control. Edit the filename to save changes to a different control file

Filename:

Number of Simulations:

Initial Noise Seed:

☐ Univariate (change one at a time)
☒ Multivariate (change all together)
☐ Latin Hypercube

Currently active parameters

sales force productivity=RANDOM_UNIFORM(200,220)
 sales force adjustment time=RANDOM_UNIFORM(3,12)
 sales person salary=RANDOM_NORMAL(1800,2200,2000,100)

Distribution

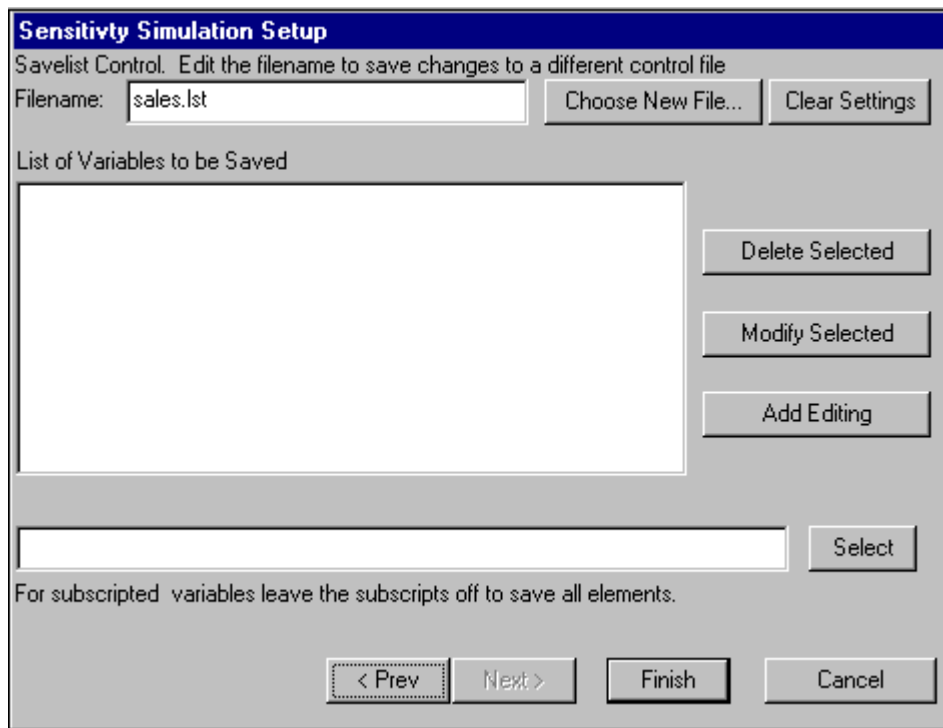
Parameter:

Model Value	Minimum Value	Maximum Value	Mean	Standard Deviation
--	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

- Click the **Next** button to move to the Save List control.

Save Lists

Save lists are files that save simulation data for any variable that is selected. Sensitivity simulations generate a huge amount of data, so it is necessary to limit the data saved to only those variables that we are really interested in. You should choose to save values only for variables that you think are of real interest; trying to save sensitivity values for all variables in the model will take a long time and require a large amount of disk space.



- ▶ Click on the **Select** button, a Variable Save dialog will open showing all model variables. Choose *cash flow* and click **OK**. Click on the button **Add Editing**.
- ▶ Click on the **Select** button and choose *Sales Force* then click **OK**. Click on the button **Add Editing**.

NOTE We could have simply typed these names individually into the editing box then clicked the button **Add Editing**.

Sensitivity Simulations

- ▶ Click on the **Finish** button.

The model will simulate once then perform 200 additional simulations while automatically varying the parameters *sales force productivity*, *sales force adjustment time*, and *sales person salary*. The dataset now contains the standard behavior for all variables with the model's original Constant values, and a range of values for the behavior of the variables *cash flow* and *Sales Force* generated by the 200 sensitivity simulations.

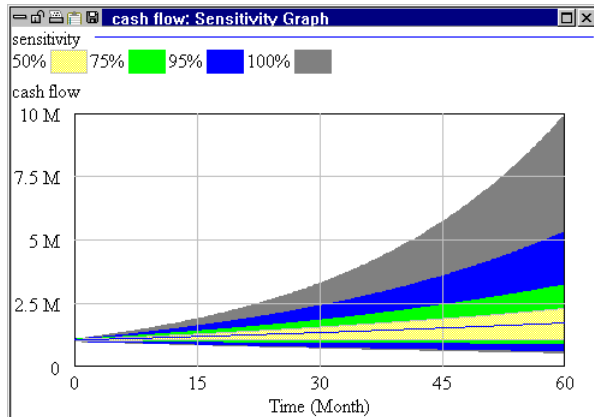
- ▶ Open the **Control Panel** and click the **Datasets** tab. Double click on the run *baserun* appearing in the **Loaded** runs list to remove it.
- ▶ Double click on the variable *cash flow* on the sketch to select it as the Workbench Variable.

Time Graph Sensitivity Output

Results of sensitivity testing can be displayed in different formats. Time graphs display behavior of a variable over a period of time. The variable's spread of values, at any period in time, are displayed either in terms of confidence bounds, or as separate values which combine to form individual simulation traces.

Confidence Bounds

- Click on the **Sensitivity Graph** Analysis tool. The default configuration for this tool is to plot confidence bounds.



A graph is generated showing confidence bounds for all the output values of *cash flow* that were generated when the three parameters were randomly varied about their distributions. You can expand the graph to full screen by clicking on the maximize/minimize button just left of the close button in the upper right corner.

The outer bounds of uncertainty (100 %) show maximum values of approximately 10 million dollars and minimum values of approximately 500 thousand dollars at the end of the simulation. Note the possibility of a decline in *cash flow*. The first simulation run (with the values of the Constants contained in the model) is plotted as a line indicated by the run name *sensitivity*. A mean value lies in between the confidence bounds, and can be plotted by:

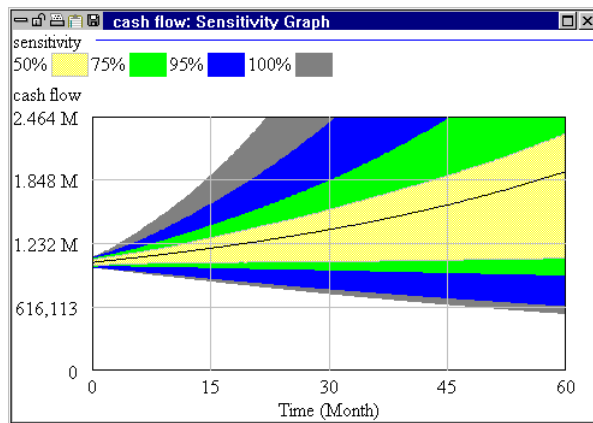
- Click with the right mouse button on the **Sensitivity Graph** tool. Click on the check box **Plot Mean Value** in the **Show Sensitivities as:** field.
- Click on the check box **Suppress first run plot**. (This will leave only the mean value and confidence bounds.) Click **OK**.
- Click on the **Sensitivity Graph** tool.

A plot is generated showing the mean value of the confidence bounds as a red line.

Focusing Graph Scales

Let's focus the vertical scale to show greater detail in the lower range of uncertainty values.

- Place the pointer on the horizontal graph line that shows 2.5 M in *cash flow*. Hold down the Ctrl key then click and hold down the mouse button. Drag the mouse so the cursor moves down to the bottom of the graph (the graph line that shows 0 in *cash flow*). Release the mouse button.
- Click on the **Sensitivity Graph** tool.



13 Using Data in Models

The Use of Data is only supported in Vensim PLE Plus. Vensim PLE will not simulate models that use data as exogenous inputs. Vensim PLE does not support data equations or the GET XLS... and GET 123... functions or importing data.

Types of Data Use

Vensim can use data in two ways. First, as exogenous inputs to drive models, and second, as a basis for comparing the behavior of a model with what actually happened.

Exogenous inputs are known as Data variables and are time series inputs which drive a portion of your model. Data variables are not computed during simulation, but instead refer to an existing data series (time series) for use during simulation. This data series can either be an imported Vensim dataset used by a **Data variable** in the model, or a time series residing in an external file such as an Excel or 123 spreadsheet used by a **Data variable** with a **Data function** in the model.

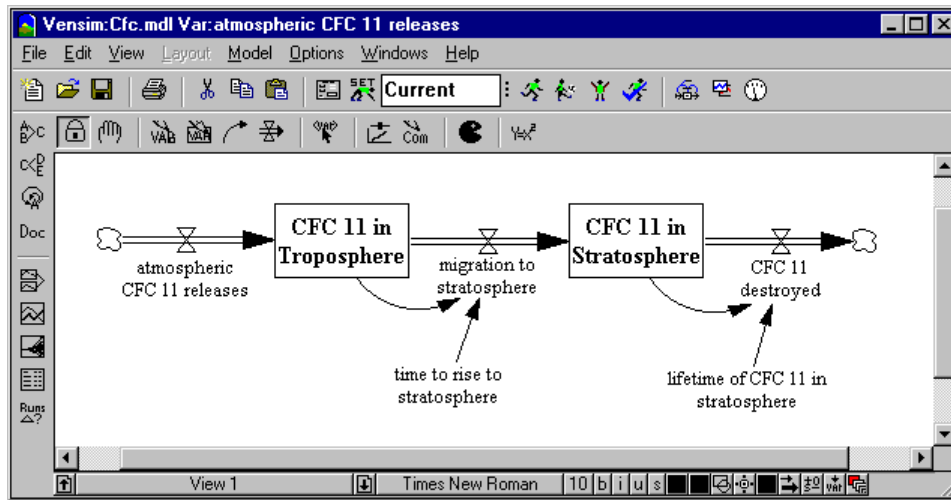
The second way to use data involves loading real world data as a Vensim dataset, then comparing the dataset against model behavior using the Analysis tools. You must have Microsoft Excel or Lotus 123 installed to complete all of this chapter. If you do not have either of these installed you can still work through much of this chapter to get an understanding of how to use data in models, but you will not be able to use the GET 123 DATA or GET XLS DATA functions.

Using Data to Drive a Model (*cfc.mdl*)

EITHER

- ▶ Open the model *cfc.mdl* in the directory *plemodel\chap13*.
- ▶ **OR** Build the model as shown in the diagram and equation listing below. Time Bounds are INITIAL TIME = 1930, FINAL TIME = 2130, TIME STEP = 0.5, Units for Time: Year.

13: Vensim PLE User's Guide



cfc.mdl Equations

$$\text{CFC 11 destroyed} = \text{CFC 11 in Stratosphere} /$$
$$\text{lifetime of CFC 11 in stratosphere}$$

Units: Mkg/Year

$$\text{CFC 11 in Stratosphere} = \text{INTEG}(\text{migration to stratosphere} - \text{CFC 11 destroyed}, 0)$$

Units: Mkg

$$\text{CFC 11 in Troposphere} = \text{INTEG}(\text{atmospheric CFC 11 releases} - \text{migration to stratosphere}, 0)$$

Units: Mkg

$$\text{lifetime of CFC 11 in stratosphere} = 55$$

Units: Year

$$\text{migration to stratosphere} = \text{CFC 11 in Troposphere} /$$
$$\text{time to rise to stratosphere}$$

Units: Mkg/Year

$$\text{time to rise to stratosphere} = 5$$

Units: Year

This model is complete except for the variable *atmospheric CFC 11 releases*.

- Save it in the directory *plemodel\chap13* with a different name (e.g., *mycfc.mdl*).

Data Variable with Data Function

A Data function resides in a model Data variable and makes a call to a spreadsheet (Microsoft Excel or Lotus 123) to retrieve the data directly.

- ▶ Select the **Equations** tool, click on the variable *atmospheric CFC 11 releases*.
- ▶ Click on the variable **Type** drop-down box (on the left side of the Equation Editor) and select the type **Data**. Click on the variable **Type** lower drop-down box (that currently says **Normal**) and select **Equation**.
- ▶ Click the **Functions** tab and in the **Function Class** drop down box, select **Data Only**. Scroll down the list of Data functions and select **GET XLS DATA** if using Microsoft Excel (or if using Lotus 123, select function **GET 123 DATA**).

The function is entered with four arguments:

1. 'filename '
2. 'tabname '
3. 'time_row_or_col '
4. 'first_data_cell '

IMPORTANT NOTE: you must enter these arguments *with* the single quotes surrounding them.

CFC emissions are included as time series data of both historical and forecast values in the directory *plemodel\chap13* in two files:

- *.xls* format (Microsoft Excel spreadsheet)
 - *.wk4* format (Lotus 123 spreadsheet)
- ▶ Type in the filename 'cfc11.xls' (don't forget the single quotes!) in the first argument if using Microsoft Excel, or 'cfc11.wk4' if using Lotus 123.
 - ▶ Double click the second argument, type in the tab name 'cfc11'.
 - ▶ Double click the third argument, type in the row name '2' (the row for time)
 - ▶ Double click the fourth argument, type in the cell name 'C4' (the starting cell for data)
 - ▶ Choose units of *Mkg/Year* and click **OK** to close the Equation Editor.

Simulating

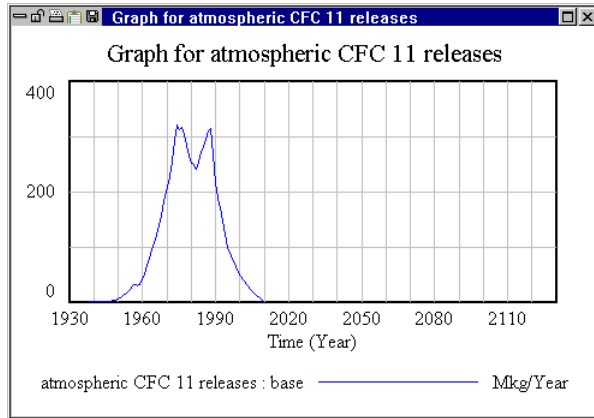
Newer versions of Excel and 123 should start automatically when called from the Vensim GET DATA function, however older versions might need to be started before you simulate. Some versions of 123 might need the *cfc11.wk4* file open in the 123 application.

- ▶ Type in a run name (e.g., *base*) and click the **Simulate** button.

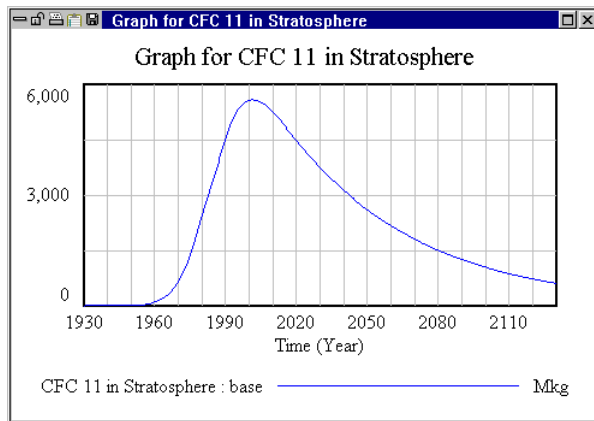
Vensim should automatically open the spreadsheet and simulate while reading values off the file *cfc11*. The model starts simulating at 1930, eight years before CFC 11 releases occurred, and Vensim reports an error because our model starts *before* the driving data. We could have entered zero in the data series for years prior to 1938, but we took the data directly from the data source.

13: Vensim PLE User's Guide

- Select *atmospheric CFC 11 releases* as the Workbench Variable and create a graph:



- Create graphs of the two levels. Note the long delay present which results in significant CFC 11 in the stratosphere for the next 100 years.



Data Variable with Imported Data

A Normal Data variable uses time series values contained in an existing Vensim dataset. This dataset must be created or imported before the model will simulate.

- Save the model *cfc.mdl* as another name, e.g. *cfc2.mdl*.
- Select the **Equations** tool, click on the variable *atmospheric CFC 11 releases*. Click on the variable **Type** lower dropdown box (currently says **Equation**) and select **Normal**. Click **OK**.

When a simulation is run, Vensim looks for a loaded dataset containing the named Data variable. If no dataset is found, the simulation is halted.

CFC emissions are included as time series data of both historical and forecast values in the directory *plemodel\chap13*. The historical data values are contained in two files:

- *.dat* format (Vensim data format text file)
- *.tab* format (tab-delimited text file)

The historical data with forecast data added are contained in two files:

- *.wk1* format (Lotus 123 spreadsheet)
- *.xls* format (Excel spreadsheet)

We will use the *.dat* data file to import a dataset into Vensim.

- Open and examine the ASCII text file *cfc11.dat* using a text editor or word processor.

Opening the file is optional, the only point is to review the data. You will see the data displayed in the format below, with the variable name followed by one column for time and one column for values:

```
atmospheric CFC 11 releases
1938      0.1
1939      0.1
(more...)
1990     216.1
1991     188.3
1992     171.1
```

- Close the *cfc11.dat* file.

Importing Text-Formatted Data (.dat)

We need to import the data series by converting the text file *cfc11.dat* to a binary Vensim data file (*cfc11.vdf*).

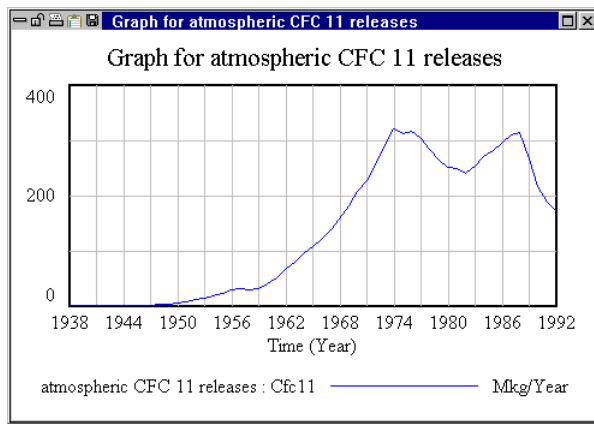
- Select the menu **Model>Import Dataset...**. Choose the file *cfc11.dat* and click **Open**.

You should get the message "Dat2vdf completed without error" and an Output window will show that a number of values were written for *atmospheric CFC 11 releases* and for time base *Time*. The dataset *cfc11.vdf* is loaded as the first dataset. Analysis tools will now work on this dataset.

- Click **OK** to the message box and close the Output window.
- Open the Control Panel and click the **Datasets** tab, double click on the previous simulation dataset *base* to remove it, leaving only the dataset *cfc11*.
- Double click the variable *atmospheric CFC 11 releases* to select it as the Workbench Variable and then click on the **Graph** tool.

A graph is displayed for *atmospheric CFC 11 releases* over the time base of the dataset.

13: Vensim PLE User's Guide



- Close the graph by clicking the Close button or pressing Del.

Simulation

Data variables in a model need data sources. These data sources are the dataset files that we imported and they need to be specified before simulation can occur. One place to set the data source is in the Simulation Control dialog; the other place is on the Toolbar after the **Set Up a Simulation** button has been pressed.

Toolbar

- Press the **Set Up a Simulation** button.
- Type in a run name (e.g., *baserun*.)
- Type in the name of the converted dataset, *cfc11*, in the left hand editing box (the Data editing box just right of the integration method **Euler**) (or press the little button right of the editing box to choose the dataset).
- Press the **Simulate** button.

Results

- Click on the **Control Panel** button, select the **Datasets** tab and double-click on the dataset *cfc11*.

Because the model used the *cfc11* dataset during simulation to provide data for one variable, the dataset *baserun* also stores the data for this variable.

If you have created your own model you should add the following Custom Graph definition:

Graph Name Hide: ☐ Title ☐ X Label ☐ Legend ☐

Title

X-Axis Sel X Label

X-min X-max X-divisions Y-divisions

Stamp Comment

☒ Norm ☐ Cum ☐ Stack ☐ Dots ☐ Lbl-Intervals Width Height

Scale	Variable	Dataset	Label	LineW	Units	Y-min	Y-max
<input type="checkbox"/>	atmospheric CFC	Sel				0	1000
<input checked="" type="checkbox"/>	CFC 11 in Tropos	Sel					
<input type="checkbox"/>	CFC 11 in Stratos	Sel					
<input type="checkbox"/>		Sel					
<input type="checkbox"/>		Sel					
<input type="checkbox"/>		Sel					

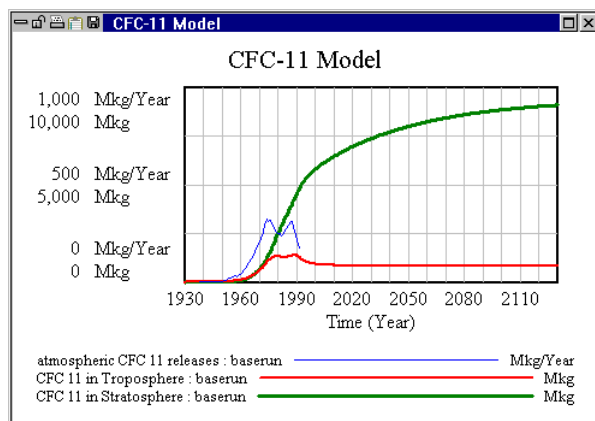
☐ As WIP Graph (maxpoints) Copy to... Test output ☐ Soft Bounds

OK Cancel

- Click the **Graphs** tab and then click on the button **Display** to show the Custom Graph.

OR

- Look at the behavior of the rates and levels in this model with the **Graph** and **Causes Strip** graph.



The input data series *atmospheric CFC 11 releases* ends at time 1992. The last figure in the series (171.1) is used in calculations for any time greater than 1992. Because of this, the *CFC 11 in Troposphere* value stabilizes after about the year 2000, while *CFC 11 in Stratosphere* keeps climbing.

13: Vensim PLE User's Guide

It is likely that the consumption and release of CFC 11 is likely to continue to decline over the following years. Releases of CFC 11 are not likely to continue at a fixed rate as high as 171.1 Mkg/Year. Let's open a data series that contains an optimistic forecast for CFC emissions.

Importing Spreadsheet Data

- If you have a spreadsheet application (e.g., Lotus 1-2-3 or Microsoft Excel), open one of the files *cfc11.wk1*, or *cfc11.xls* and examine the data.

A portion of the spreadsheet file is shown below:

Model Variables	Historical Data (see Source, below)				
	1938	1939	1940	1941	1942
atmospheric CFC 11 releases	0.1	0.1	0.1	0.1	0.1
	0.1	0.1	0.1	0.1	0.1
	0	0	0	0	0
	0	0	0	0	0

- Select the menu **Model>Import Dataset....** Choose the file *cfc11.xls* or *cfc11.wk1* and click **Open**.

A Table to VDF conversion dialog box opens. We need to set the import options for Vensim to properly read the time data, the variable data, and the variable names. First we need to set the range of cells that encompasses all the data, but nothing else.

- Click on the editing box Range: **from Row#** and type in 2.
- Click on the editing box Range: **Col#** and type in 3.
- Click on the editing box Range: **to Row#** and type in 4.
- Check that the Range: **Col#** editing box reads 61.

We do not have a variable named *Time*, so we need to specify from which row to read the time values.

- Click on the editing box for **Row#** and type in 2 (same line as **Variable is time axis**)

The option button for **Variable is time axis** should turn off and the button for **Row#** should turn on. The default position for the variable name is column #1. The model variable names are actually in column #2, so we need to specify this. We also need to exclude Row 3 of the spreadsheet because it does not contain any data.

- Click on the **Var: Col#** editing box and type in 2.
- Click on the editing box for **List of rows to exclude** and type in 3.
- Click on the button **Save Format Information...** Type in the name *format* and click the **Save** button.

You can now load this format information in the future if you want to convert the same data file again. The Table to vdf conversion dialog box should now look like:

Table to VDF conversion for: Cfc11.xls

Range ☐ All or from Row# Col# to Row# Col#

Time axis name: ☒ Across ☐ Down Time values recognized when:

☐ Variable is time axis or ☒ Row# or ☐ Formula + per col

Var: ☒ Col# or ☐ File Subs[]

☐ Strip "" Value for empty cells

List of rows to exclude: e.g. 6,9,33

List of columns to exclude:

Translation Control

Contents View

Row 1			Column 1		
Prv	Choose	Nxt	Prv	Choose	Nxt
001 Atmospheric CF			001 Atmospheric CF		
002 Model Variables			002 Year		
003 Historical Data			003 --		
004 --			004 atmospheric CF		
005 --			005 Refrigeration (M		
006 --			006 Blowing agents		
007 --			007 Open-cell foam,		
008 --			008 --		
009 --			009 --		
010 --			010 Source: Alterna		
011 --			011 xxx Washi		

- Click the OK button in the Table to VDF conversion dialog box.

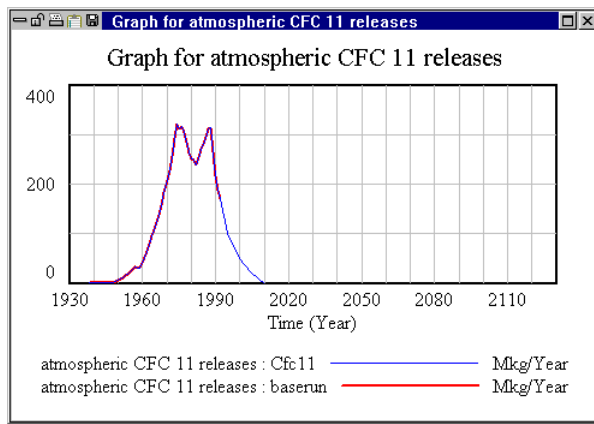
You will get the message "Dataset cfc11 already exists. Do you want to overwrite it?"

- Click **Yes**

You should get the message "Dat2vdf completed without error" and an Output window.

- Click **OK** to the message box and close the Output window.
- Double click on the variable *atmospheric CFC 11 releases* and then click on the **Graph** tool.

Now the graph of the *atmospheric CFC 11 releases* shows a drop to zero in dataset *cfc11* (dataset *baserun* has a line exactly under this, except for the values later than 1992).

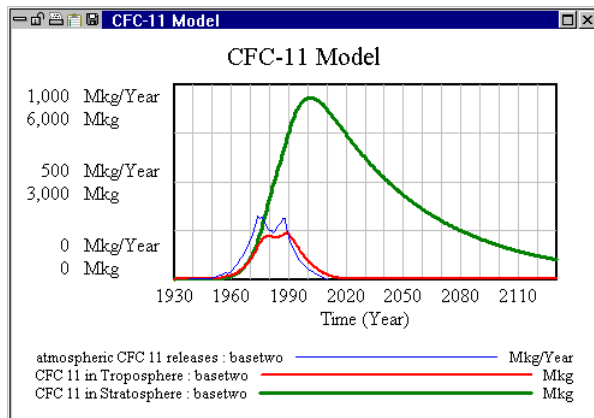


Simulation

- Click the **Set Up a Simulation** button. Type in a name for the run (e.g., *basetwo*).

Note that the dataset *cfc11* is still listed in the **Data** editing box.

- Click the **Simulate** button.
- Click the **Control Panel** button then click the **Datasets** tab and double-click on the dataset *cfc11* and the dataset *baserun* to unload them. Click the **Graphs** tab and click the button **Display**.



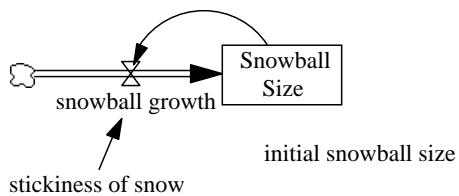
Now we see that stopping releases of CFC 11 greatly reduces the amount of *CFC 11 in Stratosphere*. However, there is a considerable delay. The peak value for *CFC 11 in Stratosphere* occurs in the year 2002, even though *atmospheric CFC 11 release* began to decline starting in the year 1988. Moreover, the value for *CFC 11 in Stratosphere* exceeds its 1988 value until about the year 2060. This model is quite simple and does not represent a complete picture of CFC release and ozone depletion. But it does demonstrate quite clearly how long it can take for the consequences of current activities to be fully realized.

These models all represent physical systems. Many of them are examples that are often formulated in simple mathematical terms.

Exponential Growth — Snowball (*snowball.mdl*)

This model demonstrates exponential growth, where the growth of something feeds upon itself. Here, the greater the snowball size, the faster it grows.

- ▶ Click the **New Model** button or select the menu item File>New Model.
- ▶ Set the Time Bounds dialog to: INITIAL TIME = 0, FINAL TIME = 10, TIME STEP = 0.25, and Units for time = Minute.
- ▶ Sketch the model shown below.
- ▶ Save your model in the directory *plemodel\chap14*. (e.g. *snowball.mdl*).



- ▶ Enter the equations below

Equations *snowball.mdl*

```
initial snowball size= 0.25
Units: Kg

snowball growth=Snowball Size * stickiness of snow
Units: Kg/Minute

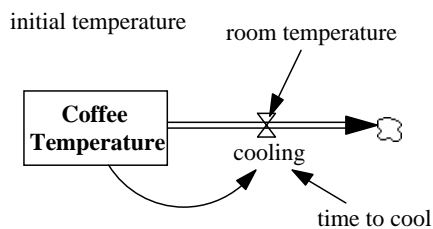
Snowball Size= INTEG(snowball growth,
                     initial snowball size)
Units: Kg

stickiness of snow= 1
Units: 1/Minute
```

Exponential Decay — Coffee Cooling (*cool.mdl*)

This model contains a first order negative feedback loop. This has the effect of pushing the system towards a goal, in other words, goal seeking.

- ▶ Click the **New Model** button or select the menu item File>New Model.
- ▶ Set the Time Bounds dialog to: INITIAL TIME = 0, FINAL TIME = 100, TIME STEP = 0.25, and Units for time = Minute.
- ▶ Sketch the model shown below.
- ▶ Save your model in the directory *plemodel\chap14*. (e.g. *cool.mdl*).



- ▶ Enter the equations below

Equations cool.mdl

```

Coffee Temperature= INTEG(
    -cooling,
    initial temperature)
Units: Farenheit

cooling=(Coffee Temperature - room temperature) / time to cool
Units: Farenheit/minute

initial temperature= 140
Units: Farenheit

room temperature=68
Units: Farenheit

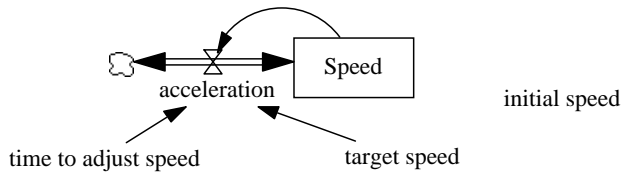
time to cool=20
Units: minute

```

Speed and Acceleration (*speed.mdl*)

Another example of a first order negative feedback loop providing goal seeking.

- ▶ Click the **New Model** button or select the menu item File>New Model.
- ▶ Set the Time Bounds dialog to: INITIAL TIME = 0, FINAL TIME = 100, TIME STEP = 0.25, and Units for time = Minute.
- ▶ Sketch the model shown below. Add an arrowhead to the rate to show a two-directional flow by right clicking on the rate handle (without the arrowhead) and checking the box for **Arrowhead**.



- ▶ Save your model in the directory *plemodel\chap14*. (e.g. *speed.mdl*).
- ▶ Enter the equations below

Equations speed.mdl

```
acceleration=
  (target speed-Speed)/time to adjust speed
Units: Miles/Hour/Second

initial speed=0
Units: Miles/Hour

Speed= INTEG(acceleration,
             initial speed)
Units: Miles/Hour

target speed=      55
Units: Miles/Hour

time to adjust speed=10
Units: Second
```

Gravitational Attraction (gravity.mdl)

The formula for the force of attraction between two bodies is given by

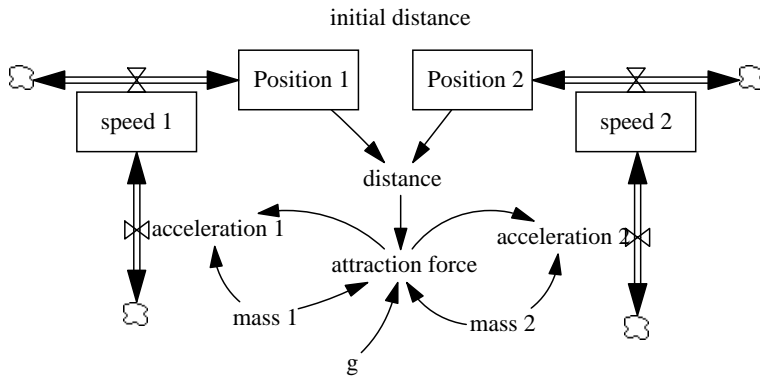
$$f = G * m_1 * m_2 / d$$

Where d is the distance between two bodies with masses m_1 and m_2 and G is the gravitational constant. This is a situation where the closer two bodies are together, the more attracted they are to one another and the closer they get. This is positive feedback and it is interesting to model this. We can represent the gravitation attraction problem in one dimension for the simple two body case.

- ▶ Click the **New Model** button or select the menu item File>New Model.
- ▶ Set the Time Bounds dialog to: INITIAL TIME = 0, FINAL TIME = 1000, TIME STEP = 5, and Units for time = Second.
- ▶ Sketch the model shown below.

The boxes around speed 1 and speed 2 are created by first naming the rates and then clicking on the names with the right mouse button (or Ctrl-Clicking on them) and selecting **Box** as the shape.

- ▶ Save your model in the directory *plemodel\chap14*. (e.g. *gravity.mdl*).



When the Equation editor is brought up for `Position 1` and `Position 2`, it will not automatically include the speed variables. This is because the box around these variables marks them as Levels, and Vensim is unsure if you mean to use them as rates as well. You can simply enter the variables into the first part of the INTEG formula. Finally the acceleration rates were created vertically and `acceleration 2` was attached to the left of the valve by right clicking (or Ctrl-click) on the valve, followed by checking **Attachment Left**.

► Enter the following equations for the model.

Equations gravity.mdl

```

acceleration 1= attraction force/mass 1
Units: Meter/Second/Second

acceleration 2= -attraction force/mass 2
Units: Meter/(Second*Second)

attraction force= g * mass 1 * mass 2 /(distance * distance)
Units: Kg*Meter/Second/Second

distance= Position 2 - Position 1
Units: Meter

g = 6.67e-011
Units: Meter*Meter*Meter/Second/Second/Kg

initial distance= 1
Units: Meter

mass 1= 10000
Units: Kg

mass 2= 1000
Units: Kg

```

14: Vensim PLE User's Guide

```
Position 1= INTEG(speed 1,0)
Units: Meter

Position 2= INTEG(speed 2,initial distance)
Units: Meter

speed 1= INTEG(acceleration 1,0)
Units: Meter/Second

speed 2 = INTEG(acceleration 2,0)
Units: Meter/Second
```

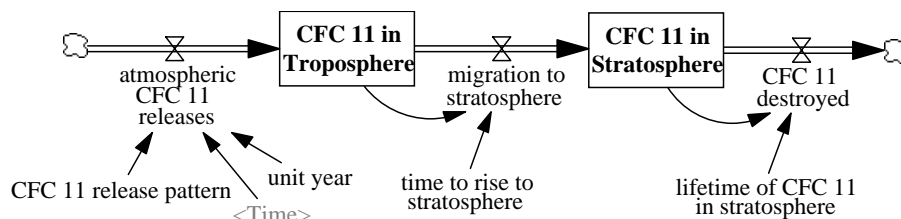
Be very careful about the signs in the different formulas, including the INTEG formulas for the positions and speeds.

The gravity model is set up so that `acceleration 2` is negative, so that the second object develops a negative speed and thus moves backward. The model has also been set up so that the two objects get closer together, but do not touch over the course of the simulation. Experimenting with mass, initial distance and the length of time the model is run should prove interesting. This model, however, does not do the right thing when the two objects get very close together or collide.

CFCs in the Atmosphere (*cfc.mdl*)

This is a simple example of cascaded negative feedback loops that demonstrates the results of CFC migration from release, through the lower atmosphere, and into the Stratosphere. This example is drawn from Miller (1995, see also Peterson 1995).

- Click the **New Model** button or select the menu item File>New Model.
- Set the Time Bounds dialog to: INITIAL TIME = 1930, FINAL TIME = 2130, TIME STEP = 2, and Units for time = Year.
- Sketch the model shown below. Note that you will need to use the **Shadow Variable** tool to add <Time>.
- Save your model in the directory *plemodel\chap14*. (e.g. *cfc.mdl*).



- Enter the equations below

Equations cfc.mdl

```

atmospheric CFC 11 releases=
    CFC 11 release pattern ( Time / unit year )
Units: Mkg/Year

CFC 11 destroyed=
    CFC 11 in Stratosphere / lifetime of CFC 11 in stratosphere
Units: Mkg/Year

CFC 11 in Stratosphere= INTEG(
    migration to stratosphere - CFC 11 destroyed,
    0)
Units: Mkg

CFC 11 in Troposphere= INTEG(
    atmospheric CFC 11 releases - migration to stratosphere,
    0)
Units: Mkg

CFC 11 release pattern([(1930,0)-(2140,400)],
    (1930,0),(1935,0),(1940,0),(1945,0),(1950,5),(1955,20),(1960,40),

    (1965,110),(1970,200),(1975,310),(1980,250),(1985,280),(1990,220),
    (1995,100),(2000,0),(2130,0))
Units: Mkg/Year

lifetime of CFC 11 in stratosphere = 55
Units: Year

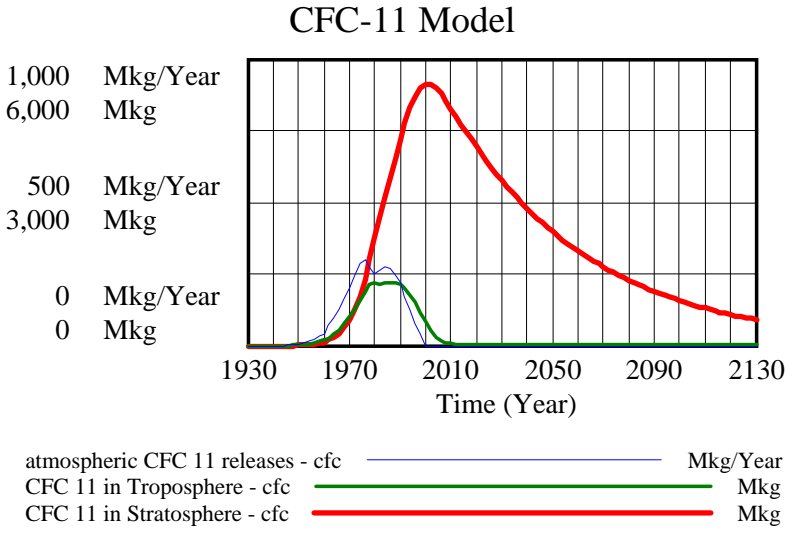
migration to stratosphere=
    CFC 11 in Troposphere / time to rise to stratosphere
Units: Mkg/Year

time to rise to stratosphere=
    5
Units: Year

unit year=
    1
Units: Year

```

One important thing to notice with this model is the delay that is introduced between the production of CFCs and their level in the Stratosphere. You can make a custom graph comparing *atmospheric CFC 11 releases*, *CFC 11 in Troposphere*, and *CFC 11 in Stratosphere* to show this delay.

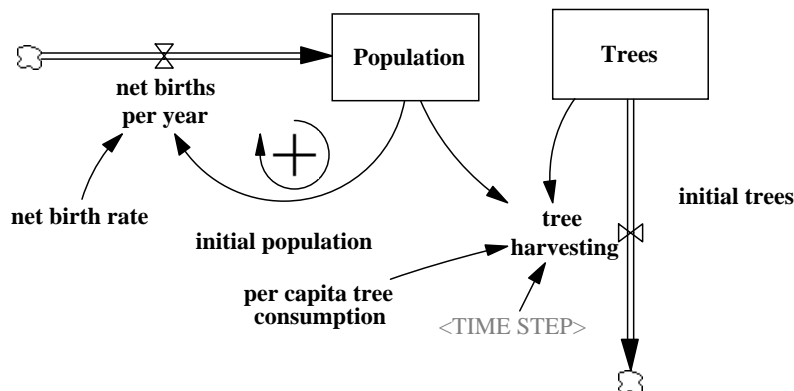


Easter Island Trees (eit.mdl)

The inhabitants of Easter island were almost completely dependent on a single species of tree for their canoes and, in turn, the majority of their food supply which was fish. A system dynamics perspective on what happened on Easter island is given in Miller (1996, see also Peterson 1996). The beginnings of that tragedy are captured in a simple model that relates the number of people and trees.

- ▶ Click the **New Model** button or select the menu item File>New Model.
- ▶ Set the Time Bounds dialog to: INITIAL TIME² = -500, FINAL TIME = 1500, TIME STEP = 10, and Units for time = Year.
- ▶ Sketch the model shown below.
- ▶ Save your model in the directory *plemodel\chap15*. (e.g. *eit.mdl*).

Easter Island Trees



You will notice in this model that `tree harvesting` depends both on the number of people - since each person needs to use trees, and on the number of trees - since if there are no trees none can be harvested. The simplest way to represent this is by saying that as long as there are trees, people continue to use them. A simple way to capture this is to write the formula

```
tree harvesting = MIN ( Population * per capita tree consumption ,
                      Trees / TIME STEP )
```

²The initial time chosen corresponds to the estimated time (500 BC) when people first landed on Easter Island. The use of a negative time, though it may seem funny at first, allows us to look at the latter part of time graphs without always having to recompute what date something corresponds to.

15: Vensim PLE User's Guide

This formula prevents the number of trees from ever going negative but otherwise lets them be consumed at the rate indicated by the number of people. The division by `TIME STEP` makes the units of measurement match up and also brings down the harvesting rate before the trees actually disappear.

► Enter the equations below.

Equations eit.mdl

```
initial population=10
Units: people

initial trees=30000
Units: trees

net birth rate=0.0035
Units: fraction/Year

net births per year= net birth rate * Population
Units: people/Year

per capita tree consumption=0.01
Units: trees/(Year*people)

Population = INTEG( net births per year,
                    initial population )
Units: people

tree harvesting=
  MIN ( Population * per capita tree consumption , Trees / TIME
        STEP )
Units: trees/Year

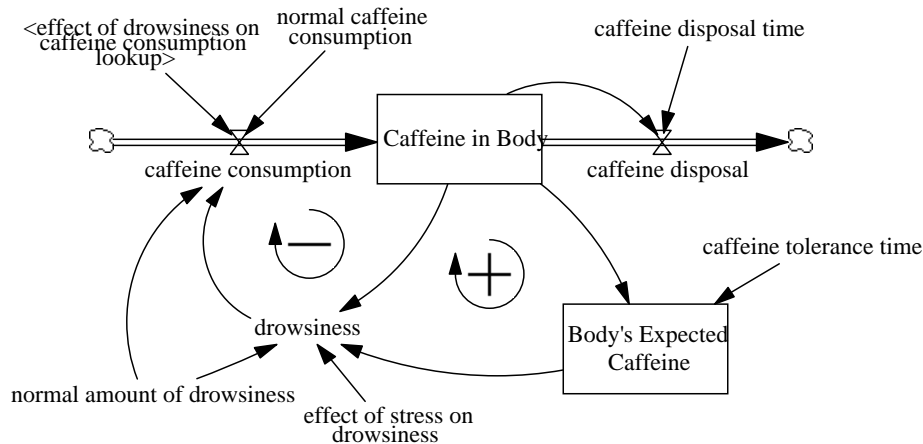
Trees= INTEG( - tree harvesting ,
              initial trees )
Units: trees
```

This model shows the roots of the problem on Easter Island, but is missing the important feedback from shortage of trees to population. You might want to reformulate this model with an explicit birth and death rate, as was done for the population model in Chapter 6, and see if you can add this feedback.

Addiction — Caffeine in the Body (*caffeine.mdl*)

The caffeine model shows how the human body grows tolerant of a substance such as caffeine, and consequently grows to depend on greater and greater quantities of the substance. The caffeine model demonstrates the dependence or addiction structure of feedback loops with different time constants. This model can be modified to show dependence on other things, both physical and psychological.

- ▶ Click the **New Model** button or select the menu item File>New Model.
- ▶ Set the Time Bounds dialog to: INITIAL TIME = 0, FINAL TIME = 50, TIME STEP = 0.0625, and Units for time = Day.
- ▶ Sketch the model shown below.
- ▶ Save your model in the directory *plemodel\chap15*. (e.g. *caffeine.mdl*).



Note that in this diagram, the level *Body's Expected Caffeine* is shown without any rates attached to it. The rate is implied in the causal connection from *Caffeine in Body*. The rates that change the level *Body's Expected Caffeine* are defined in the formula which determines the level. Using causal arrows rather than rates as sketch elements is perfectly legitimate practice and often results in a neater diagram. In this diagram, the attention is being directed to the level *Caffeine in Body* and its attached rates.

Two formulations below are new. The formula for *drowsiness* has a formulation of the form

$$= \text{XIDZ}(a, b, x)$$

In this formulation, the formula returns a / b unless $b = 0$, then it returns the value of x . This prevents divide by zero errors. The other new formulation is for *effect of stress on drowsiness*. This uses a step function to provide a test input change during the simulation, of the form

$$= x + \text{STEP}(a, b)$$

In this case, the formula returns the value of x until time equals b , then it adds a step of a to the value of x . This step input perturbs the model into producing a change of behavior.

15: Vensim PLE User's Guide

- Enter the equations below.

Equations caffeine.mdl

```
Body's Expected Caffeine = INTEG(
  (Caffeine in Body - Body's Expected Caffeine)/
    caffeine tolerance time,
  Caffeine in Body)
Units: Mg

caffeine consumption= normal caffeine consumption *
  effect of drowsiness on caffeine consumption lookup(
    drowsiness/normal amount of drowsiness)
Units: Mg/Day

caffeine disposal=Caffeine in Body/caffeine disposal time
Units: Mg/Day

caffeine disposal time= 0.25
Units: Day

Caffeine in Body= INTEG(
  caffeine consumption - caffeine disposal,
  normal caffeine consumption*caffeine disposal time)
Units: Mg

caffeine tolerance time=5
Units: Day

drowsiness=normal amount of drowsiness*
  XIDZ(Body's Expected Caffeine,Caffeine in Body,10) *
  effect of stress on drowsiness
Units: Yawn/Hour

effect of drowsiness on caffeine consumption lookup(
  (0,0.01),(1,1),(2,7),(3,9.2),(4,9.9),(5,10),(10,10))
Units: Dmnl

effect of stress on drowsiness= 1 + STEP(0.2,10)
Units: Dmnl

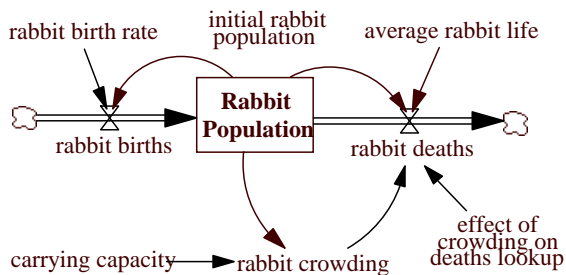
normal amount of drowsiness=1
Units: Yawn/Hour

normal caffeine consumption= 100
Units: Mg/Day
```

Rabbit Ecology (rabbit.mdl)

This model is a simple extension of the population model in Chapter 6. The rates births and deaths are set to cause exponential growth at low rabbit populations, but as the population builds it encounters crowding problems because of the limited land area (*carrying capacity*). This crowding increases the death rate, eventually canceling out the growth and balancing the two rates.

- ▶ Click the **New Model** button or select the menu item File>New Model.
- ▶ Set the Time Bounds dialog to: INITIAL TIME = 0, FINAL TIME = 5, TIME STEP = 0.25, and Units for time = Year.
- ▶ Sketch the model shown below.
- ▶ Save your model in the directory *plemodel\chap15*. (e.g. *rabbit.mdl*).



- ▶ Enter the equations below:

average rabbit life=
2

Units: Year

carrying capacity=
500

Units: Rabbit

effect of crowding on deaths lookup(
[(0,0)-(10,20)], (0,0.75),(3,2.5),(6,6),(8,11),(10,20))

Units: Dmnl

initial rabbit population=
500

Units: Rabbit

rabbit birth rate=
2

Units: fraction/Year

rabbit births=
Rabbit Population * rabbit birth rate

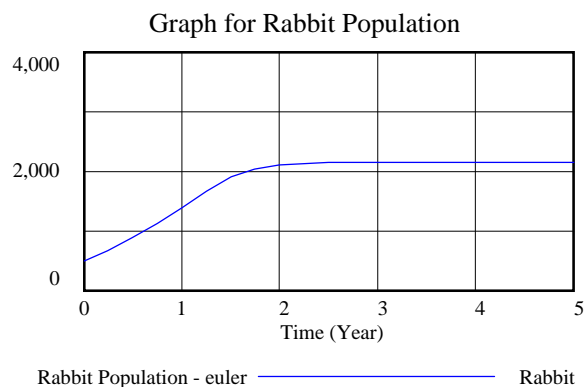
Units: Rabbit/Year

15: Vensim PLE User's Guide

```
rabbit crowding=  
  Rabbit Population/carrying capacity  
Units: Dmnl  
  
rabbit deaths=  
  Rabbit Population / average rabbit life * effect of crowding on  
  deaths lookup ( rabbit crowding )  
Units: Rabbit/Year  
  
Rabbit Population = INTEG(  
  rabbit births - rabbit deaths,  
  initial rabbit population)  
Units: Rabbit
```

► Simulate the model.

A graph of Rabbit Population shows exponential growth early on in the simulation, followed by a reduction in growth until a stable population level is reached.



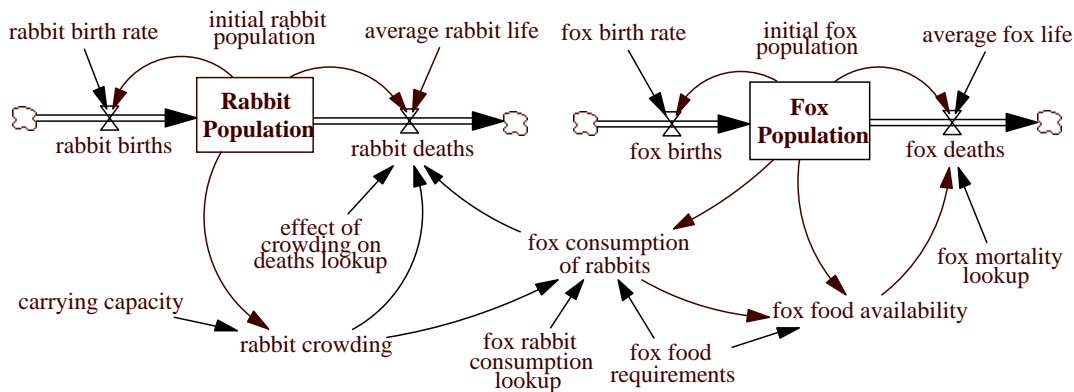
Rabbit and Fox Ecology (*rabfox.mdl*)

This model features an ecology of rabbits and foxes. This model builds on the previous model and describes the interactions between a population of rabbits (which eat grass) and foxes (whom eat rabbits). This model is relatively simple, by standards of models that are often constructed, but generates interesting and insightful dynamic behavior.

- Click the **New Model** button or select the menu item File>New Model.
- Set the Time Bounds dialog to: INITIAL TIME = 0, FINAL TIME = 50, TIME STEP = 0.25, and Units for time = Year.

Note the increased time horizon from the previous rabbit model.

- Sketch the model shown below.
- Save your model in the directory *plemodel\chap15*. (e.g. *rabfox.mdl*).



- Enter the equations below:

Equations *rabfox.mdl*

average fox life=
4

Units: Year

average rabbit life=
2

Units: Year

carrying capacity = 500

Units: Rabbit

effect of crowding on deaths lookup(
[(0,0)-(10,20)], (0,0.75),(3,2.5),(6,6),(8,11),(10,20))

Units: Dmnl

15: Vensim PLE User's Guide

```
fox birth rate = 0.25
Units: fraction/Year

fox births = Fox Population * fox birth rate
Units: Fox/Year

fox consumption of rabbits=
  Fox Population * fox food requirements * fox rabbit consumption
  lookup(rabbit crowding)
Units: Rabbit/Year

fox deaths=
  Fox Population / average fox life * fox mortality lookup(fox food
  availability)
Units: Fox/Year

fox food availability=
  fox consumption of rabbits/(Fox Population * fox food
  requirements)
Units: Dmnl

fox food requirements = 25
Units: Rabbit/Year/Fox

fox mortality lookup
  [(0,0)-(2,20)], (0,20),(0.3,5),(0.5,2),(1,1),(2,0.5))
Units: Dmnl

Fox Population= INTEG(
  fox births - fox deaths,
  initial fox population)
Units: Fox

fox rabbit consumption lookup
  [(0,0)-(6,6)],(0,0),(1,1),(2,2),(6,2) )
Units: Dmnl

initial fox population=
  30
Units: Fox

initial rabbit population=
  500
Units: Rabbit

rabbit birth rate=
  2
Units: fraction/Year

rabbit births=
  Rabbit Population * rabbit birth rate
Units: Rabbit/Year
```

```

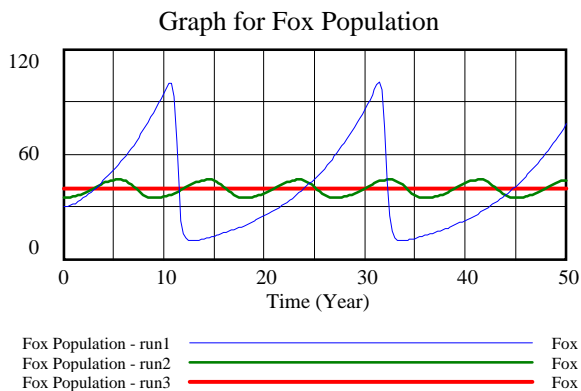
rabbit crowding=
  Rabbit Population/carrying capacity
Units: Dmnl
rabbit deaths=
  MAX(Rabbit Population / average rabbit life *
    effect of crowding on deaths lookup(rabbit crowding),
    fox consumption of rabbits)
Units: Rabbit/Year
Rabbit Population= INTEG(
  rabbit births - rabbit deaths,
  initial rabbit population)
Units: Rabbit

```

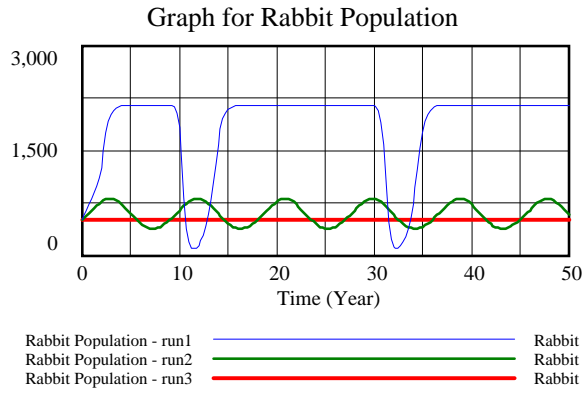
Simulation, Analysis, Experiments

- Click on the **Set Up a Simulation** button and choose the integration technique Runge Kutta 4 Auto. This model will not simulate properly with Euler integration. Type in a run name and click the **Simulate** button.
- Investigate the behavior of the Rabbit and Fox populations. Compare them separately, or create a Custom Graph to show both variables on one graph.
- Experiment by running simulations with values for *initial fox population* of 35 and 40 foxes.

Compare the three different simulation runs.



15: Vensim PLE User's Guide



The first run (*initial fox population = 30*) displays growth, stability, and collapse in Rabbit Population, with growth and collapse in Fox Population. The second run (*initial fox population = 35*) displays sustained oscillation in both Rabbit and Fox Populations. The third run (*initial fox population = 40*) shows perfect equilibrium in both Rabbit and Fox Populations.

16 Workforce, Inventory and Oscillation

This chapter is focused on the conceptual issues in the development, analysis and use of a model. After working through this chapter you should have a better understanding of how to think about, and work through, a dynamic model with Vensim.

Background

You are involved in the production and sale of prefabricated window frames. Overall your company is doing quite well, but you often go through periods of low capacity utilization followed by production ramp up and added shifts. While all of this is normally blamed on market demand and the condition of the economy, you have your doubts. Looking back at sales and production over the last 8 years it seems that sales is more stable than production. Your goal is to determine why this might be, and what you can do about it.

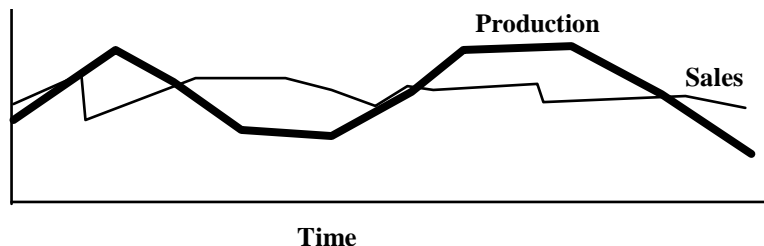
In attacking this problem you want to simplify as much as possible your current situation. There are a number of reasons for this simplification:

- It is easier to understand a simple model.
- You can get results quickly and decide if you are on the right track.
- It is more effective to start with a simple model and add detail, than to build a complex model and attempt to extract insights from it after it is complete.
- Using a simple model forces you to take an overview which is usually useful in the initial modeling phases.

It is not always true that you want to start building simple models. In many cases the behavior you see is the result of complexity, and Vensim provides a very rich set of tools for dealing with complexity. Until you have had substantial experience, however, simplicity is highly recommended. While it is not uncommon to discover that what you have developed is not rich enough for the problem you wish to address, it is rare not to gain understanding in the process. Conversely, large complex models can become significant resource drains, providing no payoff for a very long time.

Reference Modes

A reference mode is a graphical statement about a problem. Verbally, the problem was stated as "production is less stable than sales." Graphically we might draw:



This reference mode is a sketch of behavior we might expect a model to produce. It might be real data from your records, or your expectation of what might happen in a new situation. The reference mode is used to focus activity. Having mapped out one or more reference modes the goal is to define the simplest structure that is sensible³ and capable of generating patterns of behavior that qualitatively resemble the reference modes. If appropriate, such a structure can also be refined in order to develop a model that can be validated quantitatively against the available data.

Dynamic Hypothesis

A dynamic hypothesis is an idea about what structure might be capable of generating behavior like that in the reference modes. For this example we can formulate a dynamic hypothesis simply by thinking about how the two variables in the reference mode are connected — that is by specifying the set of policies (or rules) that determine *production* given *sales*. The dynamic hypothesis for this firm is that a manager is setting production based on current sales, but is amplifying the amount resulting in higher (or lower) production than is necessary. The reference mode supplies us with two variables — *production* and *sales* — that we will want to include in the model. This is a reasonably good basis on which to begin a sketch, so let us put these variables down to start the model.

Workforce / Inventory Model (*wfinv1.mdl*)

- ▶ Click the **New Model** button or select the menu item File>New Model.
- ▶ Set the Time Bounds dialog to: INITIAL TIME = 0, FINAL TIME = 100, TIME STEP = 0.25, and Units for time = Month.
- ▶ Save your model (e.g., *wfinv1.mdl*) in the *plemodel\chap16* directory.

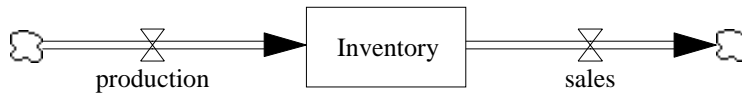
Add the variables *production* and *sales* to the sketch. Now the question arises, how are *production* and *sales* related. Clearly there is a close relationship, since it is necessary to produce something before it is sold. Sales and production are related in two ways:

1. Physical: production is required to produce goods to sell
2. Information: managers base production decisions on current or recent sales

We will start the model with the physical side. When production occurs, goods are not immediately sold. Instead, they are stored in an inventory until a sale occurs, at which point they are removed from inventory. In general, there will be an inventory, or some combination of inventory and backlog, separating production from sales. If a backlog is used in the model, it is useful to consider orders and shipments instead of simply sales. In this model, we will just use an inventory.

We construct *Inventory* as a Level, then add a rate flowing in and a rate flowing out. Next, we use the variable merge tool to drag our two existing variables, *production* and *sales* onto the valves.

³The development of a feedback model is not so much the search for correct structure as the avoidance of structure that is directly in contradiction with reality.

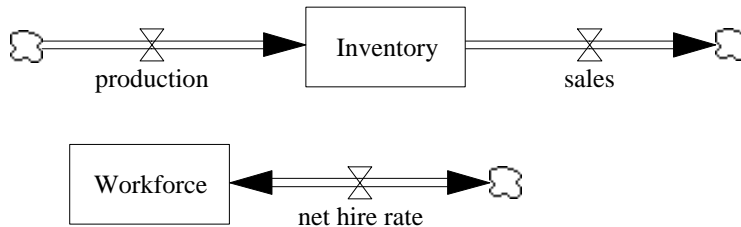


It is worth noting at this point that we could have created the same diagram by first entering the level containing *Inventory*, then adding and naming the two rates. The reason we chose to add *Inventory* then attach the existing variables as rates was to work through the problem as it came to light, rather than working out the problem first then putting it on the sketch.

Workforce

Now we need to figure out how *production* gets determined. Over the long term investment and capacity are clearly important, but these have been extremely stable. In the shorter term more people are hired and, if necessary, an additional production shift added. There is quite a bit of complicated stuff going on when shifts are added: management changes, maintenance scheduling problems arise, and so on. However, as a first approximation, more people make more products, and this is a good starting point, so we add the level *Workforce*.

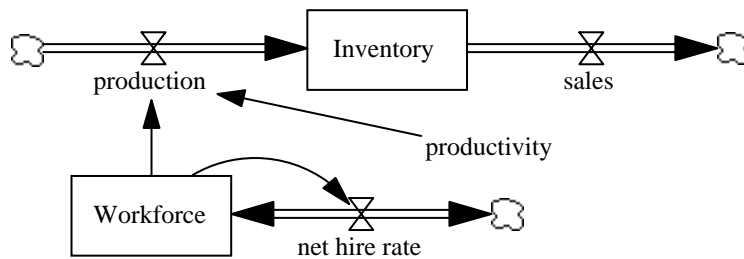
The things that change workforce are hiring, layoffs, firings and retirements. Again, for simplicity we combine all of these into a composite concept — the *net hire rate*. Note that net hire rate can either increase or decrease the workforce.



Behavioral Relationships

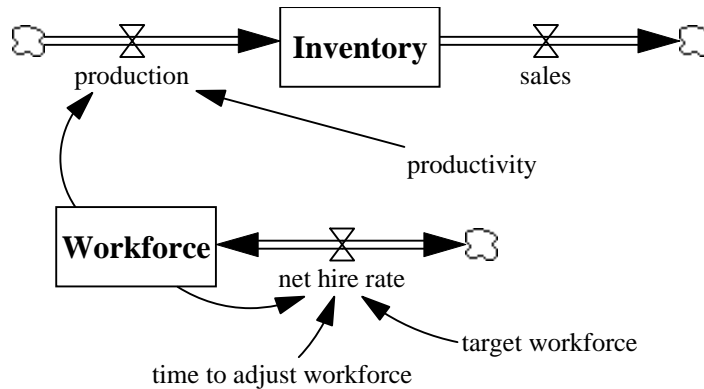
This is the physical part of the problem. Now it is necessary to make some of the behavioral (information) connections. Putting down the important physical stocks and flows is often a good starting point in developing a model. It lets you make part of the system concrete, and this can simplify the conceptualization of other parts of the system. Alternative approaches include building a causal loop diagram and converting that to stock and flow form, or writing equations directly from causal loop diagrams. You might also want to draw a causal loop diagram, then start over again with a stock and flow diagram. What works best varies by individual and by problem, so we try to present some alternative approaches in different chapters in this guide.

In completing the information connection, we will try to keep things as simple as possible. Starting with production we want to remove all the complexities of adding shifts and mothballing equipment and simply state that *production* is proportional to *Workforce*. We add the proportionality constant *productivity*. Also, *net hire rate* is dependent on the value of *Workforce*. This gives us:

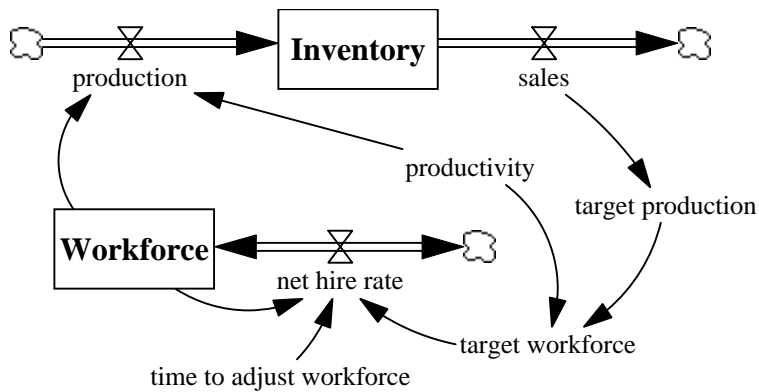


The net hire rate is the net number of people hired. The most straightforward way to formulate this is as a stock adjustment process. In a stock adjustment process you take an existing value of a variable (usually a stock) and compare it to some target or desired level, then take an action based on the difference between the two. For example if you are driving a car at 40 MPH and wish to be going 50 MPH you would depress the accelerator. Your car's speed (a stock) will increase at a rate that depends on how far you depress the accelerator and the car you are driving.

To capture this stock adjustment process we add in the variables *target workforce* and *time to adjust workforce* and connect them as shown :



time to adjust workforce represents the time required for management to agree on a change in the workforce level and screen potential applicants or notify workers to be laid off. *target workforce* is the number of people you need to produce the amount you want to produce. The Level *Workforce* is initialized at this value. Now we add the concept of *target production*, and connect it to *target workforce*. We will set *target production* on the basis of *sales*.



This is a complete model, though it does have a critical error of omission which will be brought to light during simulation. The next step is to take the conceptual model and turn it into a simulation model.

Equation Set *wfinv1.mdl*

The full equation set for this model follows:

```

FINAL TIME = 100
Units: Month

INITIAL TIME = 0
Units: Month

TIME STEP = 0.25
Units: Month

SAVEPER = TIME STEP
Units: Month

Inventory = INTEG(production-sales,
300)
Units: Frame

net hire rate=(target workforce-Workforce)/time to adjust workforce
Units: Person/Month

production= Workforce*productivity
Units: Frame/Month

productivity= 1
Units: Frame/Month/Person

sales= 100 + STEP(50,20)
Units: Frame/Month

target production= sales
Units: Frame/Month

```

16: Vensim PLE User's Guide

```
target workforce= target production/productivity
Units: Person

time to adjust workforce= 3
Units: Month

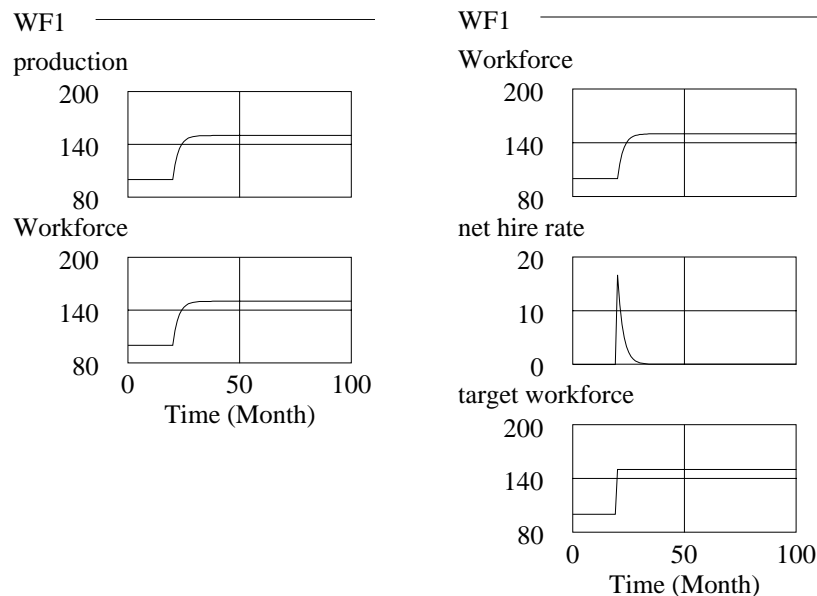
Workforce = INTEG(net hire rate,
                  target workforce)
Units: Person
```

Each equation is consistent with discussion during the initial conceptualization. The equation for *sales* has *sales* steady at 100 until time 20, when *sales* step up to, and thereafter remain at, 150. This input pattern is used to test that the system is indeed at an equilibrium, and then check the adjustment to a new operating condition. This step input pattern is the cleanest pattern available for looking at the internally generated dynamics of a system such as this. It allows you to observe how a single shock propagates through the system without further external disturbance.

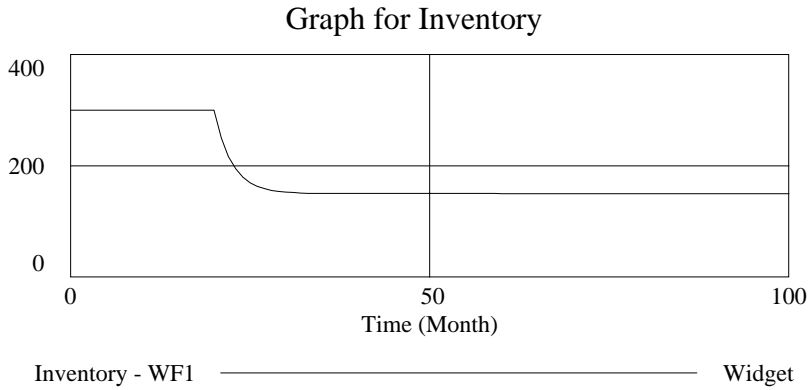
The model is complete. A simulation is performed with the name WF1.

Analysis

A Causes Strip graph for *production* and causes strip of *Workforce* show the dynamics.



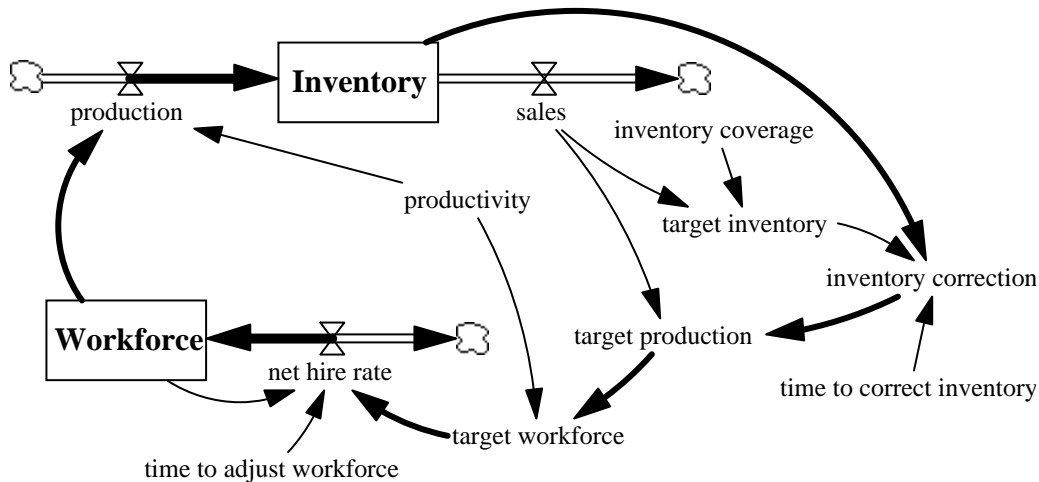
It is immediately obvious that there is not a lot of variability in *production* or *Workforce*. There is very smooth adjustment from the initial 100 Widgets/Month to 150 Widgets/Month. Unlike our reference mode, the model does not appear to generate more variability in production than sales. At this point it is worth taking a look at *Inventory*.



Inventory falls smoothly from its initial value of 300, to about 150. Since the purpose of holding an inventory is to be sure that the right product configuration is available for customers, there is clearly something wrong. Any discrepancy in *Inventory* from the level necessary to meet product mix requirements and have a comfortable safety stock needs to be corrected, and this model does not make that correction.

Model Refinement (*wfinv2.mdl*)

In order to refine the model we introduce *target inventory*, *inventory correction* and two additional Constants. The idea is simple — *target inventory* is the amount of stock that should be held based on expectations about sales. The *inventory correction* is the correction for a deviation of *Inventory* from its target. A new loop has been introduced and is highlighted.



Additional Equations

target production = *sales* + *inventory correction*
Units: Frame/Month

inventory correction = (*target inventory* - *Inventory*)/
 time to correct inventory
Units: Frame/Month

time to correct inventory = 2
Units: Month

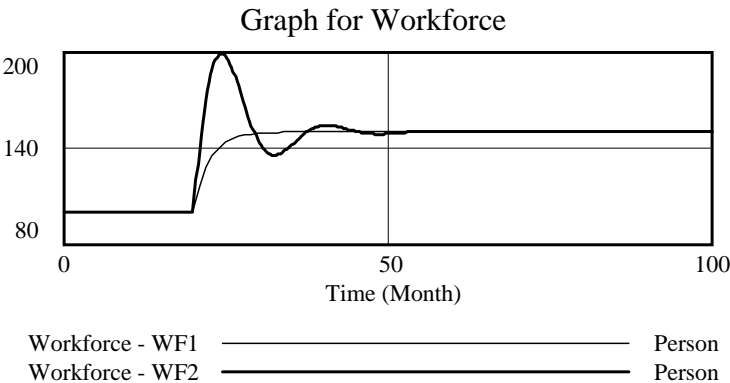
target inventory = *sales* * *INVENTORY COVERAGE*
Units: Frame
INVENTORY COVERAGE = 3
Units: Month

inventory correction is a stock adjustment formulation, just as *net hire rate* was. The *time to correct inventory* represents the time required to notice significant changes in inventory and schedule corrections in production.

The important difference between this formulation and that of *net hire rate* is that the *net hire rate* directly influences the stock it is attempting to adjust (*Workforce*) whereas *inventory correction* influences *target production*, *net hire rate*, *Workforce*, *production* and finally *inventory*. This connection has an intervening level, *Workforce*, which has important dynamic consequences.

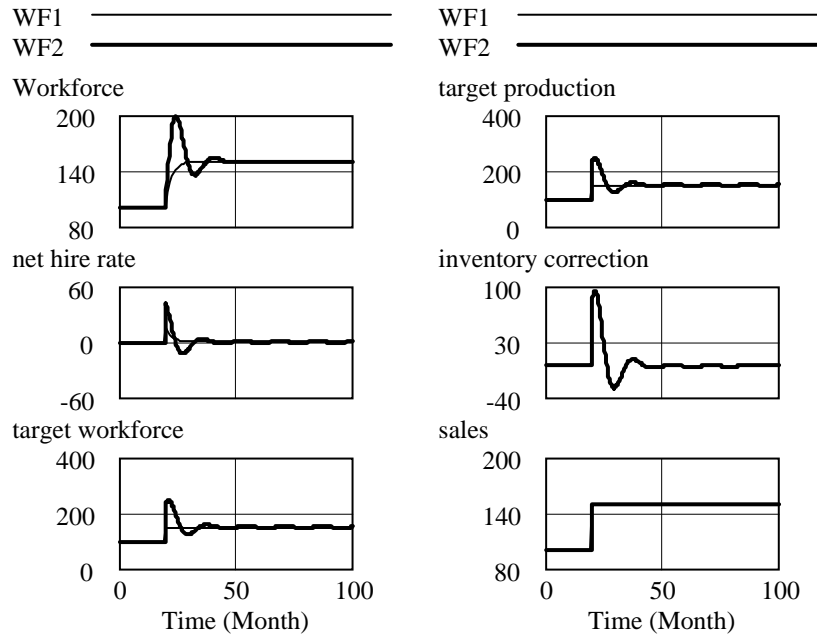
Refined Model Behavior

The model is simulated and the run named WF2. First generate a graph of behavior for *Workforce* with datasets from both runs loaded WF1 and WF2.

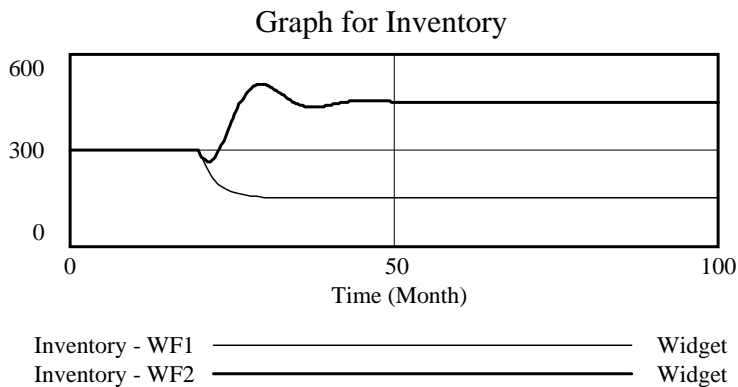


16: Workforce, Inventory and Oscillation

The behavior is dramatically different from the earlier model. *Workforce* is less stable and there is oscillation. Causal tracing is performed on *Workforce*, and on *target workforce* (graph not shown) and *target production*, and the output shown below:

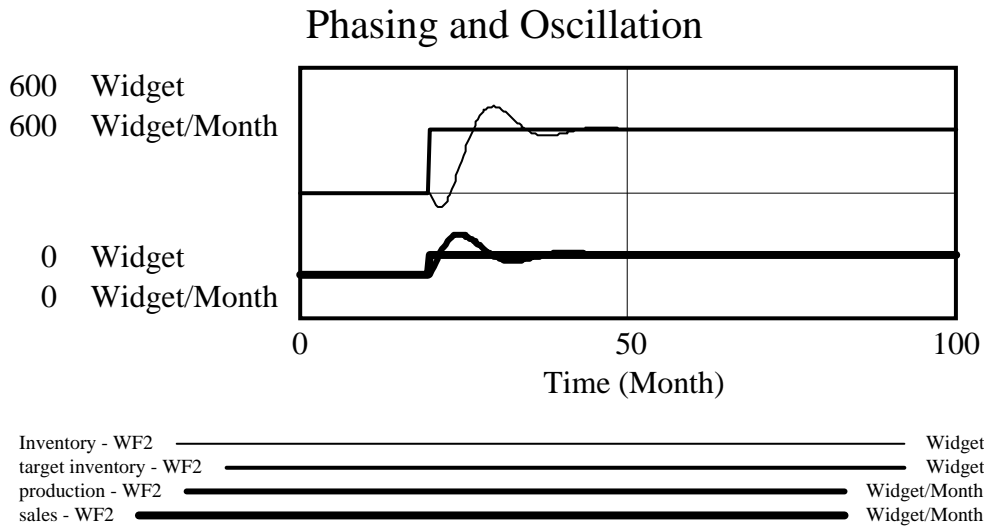


inventory correction was not in the first model. Therefore there is no plot from WF1 for *inventory correction*. All the oscillatory behavior in *target production* is due to *inventory correction*. A graph of *Inventory* shows similar oscillation, but a different final value from the first run. When sales increase, inventory now eventually adjusts to a new, higher desired level, rather than falling as before.

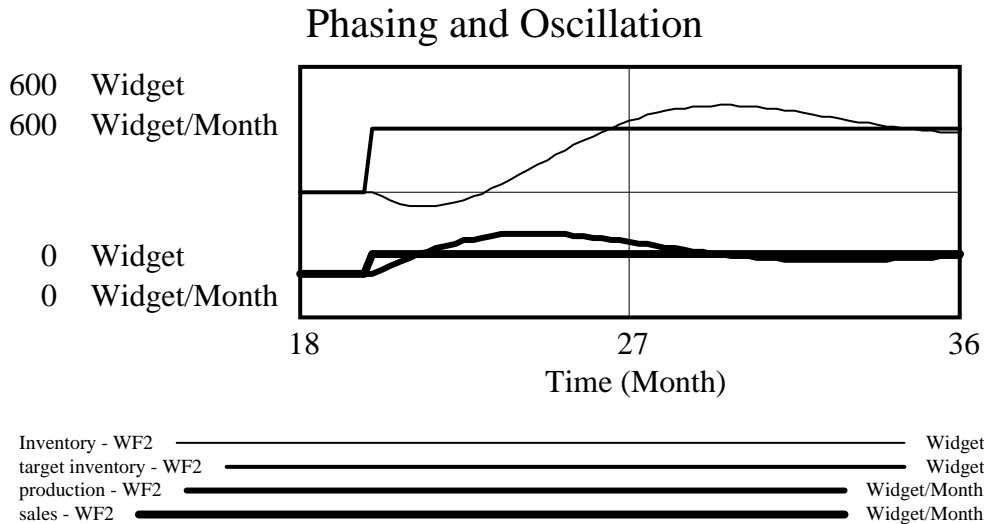


Phasing and Oscillation

To get some useful insight into this model, and oscillations in general, use the Graphs tab in the Control Panel to create a graph containing *Inventory*, *target inventory*, *production* and *sales*. Set the scales to zero minimum and 600 maximum values for all variables. You will also need to go the Datasets tab of the Control Panel and put WF2 first.



We want to focus in on the timing of changes just after the jump in sales. Hold down the shift key and dragging across the portion of the graph of interest, or go to the Time Axis tab of the Control panel, to set the time focus from approximately 18 months to 36 months. Now the custom graph generates this output:



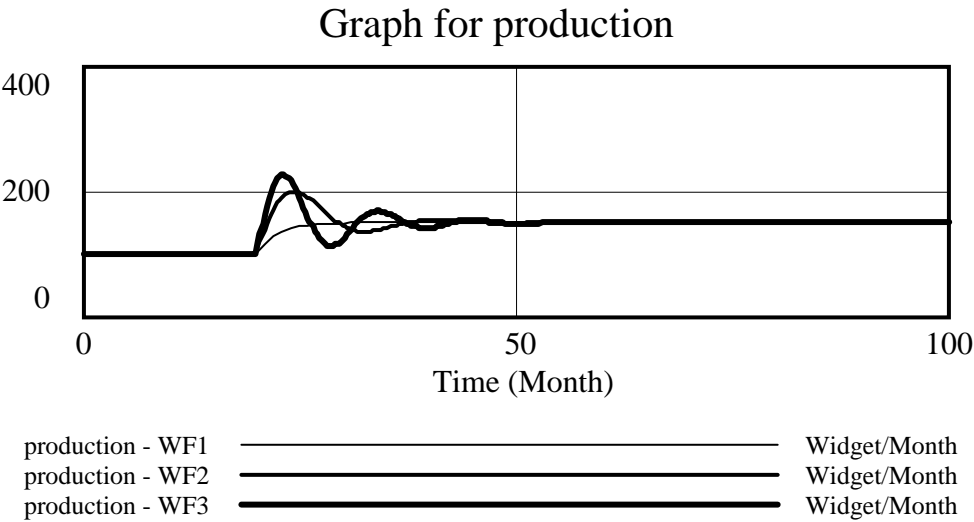
Initially, when *sales* step up, inventory begins to fall because *sales* exceed *production*. *production* gradually increases because *target production* increases with *sales* and *inventory correction* increases as inventory falls. However, there is a delay in this process due to the time required to hire new workers. By month 21.3, *production* is equal to *sales*, so *inventory* stops falling and then begins to grow.

While *Inventory* is increasing, so is *production*. As long as *inventory* is below *target inventory*, there is pressure to increase *production*, even when ***production*** is already above *sales*. In fact, *production* needs to get sufficiently above *sales* so that the ongoing difference balances the pressure from *inventory correction* before *net hire rate* will go negative. Note that when *Inventory* is equal to *target inventory*, *production* is still higher than *sales* because the workforce is excessive, causing an ongoing increase in *Inventory*. The variability in *production* is now greater than that in *sales*; the model is exhibiting the kind of amplification we identified in the reference mode.

In fact, the amplification of variability from sales to production is physically inevitable. The change in production must exceed the change in sales for some time in order to replace inventory lost before production can adjust and to adjust inventory to the new, higher target level.

Sensitivity

It is interesting to experiment with the model by changing different parameters. One possible policy to improve the company's performance would be to correct deviations in inventory more aggressively. We can simulate this by using a decreased value for *time to correct inventory*. Reducing *time to correct inventory* from 2 months to 1 month, we get the simulation output for *production*:



Here we find that reducing the delay in correcting inventory actually increases the amplitude (and decreases the period) of the oscillations, not necessarily a desirable thing for a firm to experience!

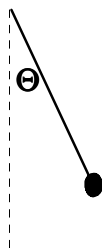
Having worked through the modeling exercises the explanation for this response is quite intuitive. The system needs to overshoot in order to build inventory. Trying to rebuild inventory more quickly will increase the size of that overshoot. Because of the delays involved that overshoot will also lead to an overbuilding of inventory which will require a decrease in production. Because the overshoot is bigger, the decrease will also be bigger.

17 Pendulums and Oscillation

In Chapters 3 and 16, we examined a model of the Workforce-Inventory oscillator. Although useful in its own right, that model is most important for the insights it provides into the process of oscillation. In this chapter we review a model of an undamped pendulum. This chapter describes the integration technique of Runge-Kutta (Variable). This integration method provides more accurate results than Euler integration when the model is an undamped oscillator and/or is poised on the edge of instability.

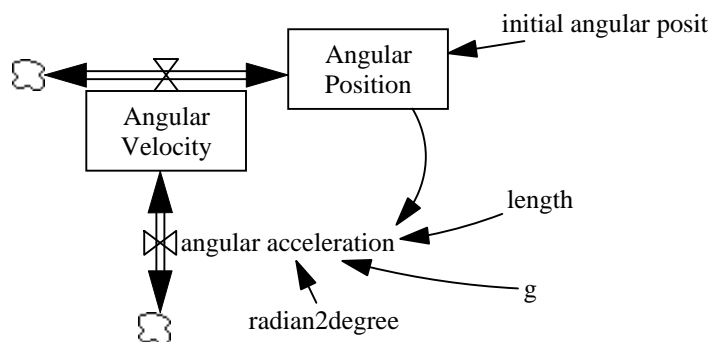
The Pendulum (*pendulum.mdl*)

A pendulum consists of a weight attached to a string suspended from a hook or other solid attachment. It is a common and well understood device that has been in used in clocks for several centuries. The reason that pendulums are valuable for time keeping is that they maintain a relatively constant period so long as their range of motion is small.



If we let Θ represent the deviation of the pendulum from straight up and down, then the force on the pendulum can be decomposed into a component parallel to the string ($mg \cos(\Theta)$, where m is the pendulum mass and g is the acceleration of gravity) and a component perpendicular to the string ($mg \sin(\Theta)$). Since the weight can only move perpendicular to the string, the accelerating force is perpendicular to the string, so that the linear acceleration is just $g \sin(\Theta)$. We can convert this linear acceleration to a radial acceleration by dividing by the length of the string.

We represent this physical problem with the diagram:



17: Vensim PLE User's Guide

Notice the cascaded levels. The acceleration of an object changes its velocity, which in turn changes its position. This means that there is a rate which is also a level. Although this makes good sense for this example, such a situation is quite rare for nonphysical models. In the Workforce-Inventory example, *Inventory* is analogous to *Angular Position*, and *Workforce* is analogous to *Angular Acceleration*. In that case, however, *Workforce* does not directly cumulate into *Inventory*, but is instead responsible for a process (*production*) that cumulates into *Inventory*, all of which introduces more loops as discussed in Chapter 16. This model has only a single feedback loop.

For simplicity, this model uses angles computed in degrees. *radian2degree* is a constant ($=360/2\pi$) that converts between radians and degrees, and is used in the computation of $\sin(\Theta)$. To maintain dimensional consistency within the equations the unit of measure Radian is considered to be equivalent to dimensionless.

- ▶ Click the **New Model** button or select the menu item File>New Model.
- ▶ Set the Time Bounds dialog to: INITIAL TIME = 0, FINAL TIME = 10, TIME STEP = 0.03125 seconds, and Units for time = Second.
- ▶ Sketch the model shown above.
- ▶ Save your model (e.g. *pendulum.mdl*) in the *plemodel\chap17* directory.

The equations for this model are:

```
Angular Position = INTEG(Angular Velocity,initial angular position)
Units: Degree
```

```
Angular Velocity = INTEG(angular acceleration,0)
Units: Degree/Second
```

```
angular acceleration = -radian2degree*
    SIN(Angular Position/radian2degree) * g / length
Units: Degree/Second/Second
```

In this equation you will notice that *Angular Position* is converted from degrees to radians by dividing by *radian2degree*. The result ($-\sin(\Theta) g/\text{length}$) is then in radians and is converted back to degrees by multiplying by *radian2degree*.

```
g = 9.2
Units: Meter/Second/Second
```

```
initial angular position = 20
Units: Degree
```

```
length = 0.5
Units: Meter
```

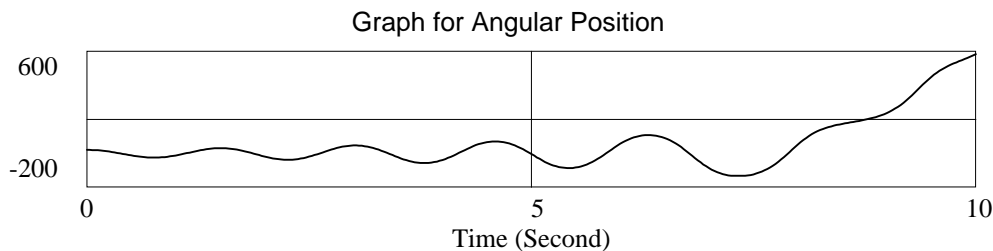
```
radian2degree=57.296
Units: Degree
```

Simulating the Pendulum

If you have ever solved the equations of motion for a pendulum all of the above should seem very familiar with one exception. Normally the simplifying assumption $\sin(\Theta) = \Theta$ is made. Because we are simulating these equations, rather than solving them explicitly, the simplification is not necessary. In fact, we can use this model to understand the implications of larger swings on the period of the pendulum, something that is not possible if the equations are simplified.

Euler Integration

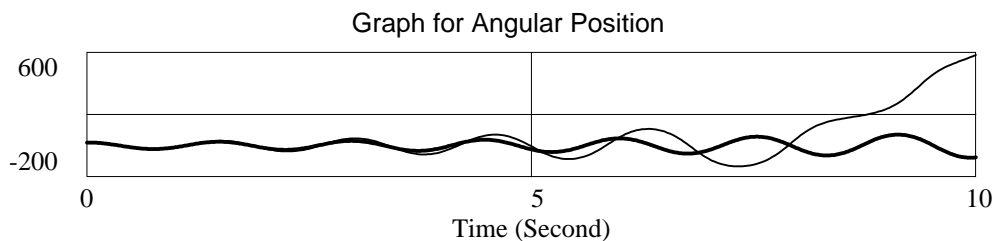
If you attempt to simulate this model using Euler integration, you will get the surprising result:



Angular Position - Euler ————— Degree

If you are puzzled, this graph says that if you raise the weight to 20 degrees and let go, the pendulum goes back and forth a few times, and then begins to spin around its axis (angular position $> 360^\circ$). While this may be consistent with a child's notion of how a swing ought to work, you would be hard pressed to design a pendulum to do this. Something is wrong, and it has to do with integration.

When you are using Euler integration and suspect that it is giving bad results, a good test is to cut the integration period in half, and see if the results change. We can do this by making the simulation run Euler and changing `TIME STEP` to `.015625`:



Angular Position - Euler ————— Degree
 Angular Position - Euler2 ————— Degree

The results change a great deal, but still do not make sense. The oscillations are still growing, and rotation would likely result in a few more seconds.

17: Vensim PLE User's Guide

In order to get good results from this model using Euler integration you need to make `TIME STEP` very small indeed ($< .001$). If you do want to try this be sure to change `SAVEPER` to be a number (.0625 or .03125) or Vensim PLE will try to store 10,000 values for each variable and, most likely, fail to do so. There is, however, a better solution.

The reason that Euler integration fails for this model is that this model represents an undamped oscillator on the edge between stability and instability. Euler integration is a simple linear extrapolation method, and when you try to extrapolate for something on a curve, you always overshoot the turning point. Normally a small overshoot is not much cause for concern, but in this case it provides a little bit extra displacement on each cycle, and that adds up to give complete divergence.

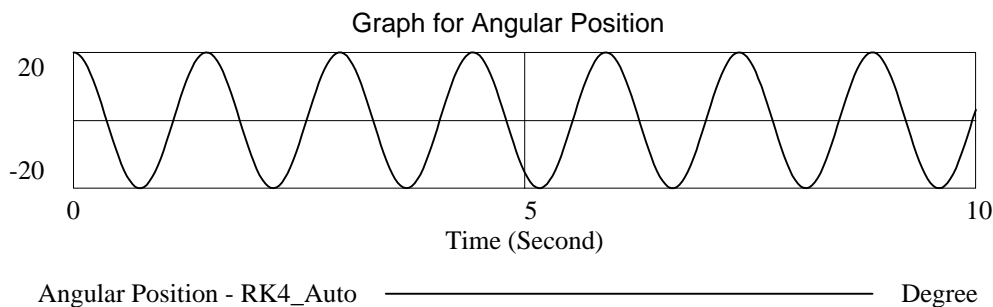
Runge-Kutta Integration

Vensim PLE provides an alternative integration technique for dealing with models such as this. This methods, referred to as Runge-Kutta Integration, provide higher order extrapolation, looking at both the trajectory and how the trajectory is changing to give a better solution.

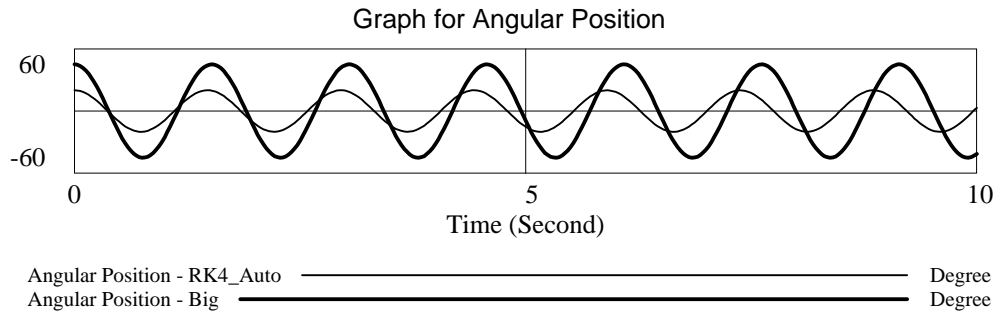
When you click on the **Set Up Simulation** button you will notice that the name Euler appears at the left. Click on this and it will change to RK4 indicating that the integration technique will be 4th order Runge Kutta (with automatic adjustment of the step size). This integration technique performs, automatically, the experiment of decreasing `TIME STEP` (as done above) to check accuracy. If accuracy is below an acceptable tolerance, the integration interval is decreased further until the desired accuracy is obtained. This is not always possible, and you will sometimes receive warning messages about being unable to achieve the desired accuracy.

It is important to note that when you are using the Runge-Kutta integration techniques, you will not see all intermediate computations. `TIME STEP`, and therefore normally `SAVEPER`, are not affected by the integration technique. With any of the Runge-Kutta integration techniques there will always be computations made between `TIME STEPS`. You can, consequently, see a level that does not seem to be the accumulation of its inflows and outflows. For truly continuous systems this will not matter, but if there is any switching going on things can get very confusing.

- Select a new run name (e.g. RK4_Auto).
- Click on the **Set Up Simulation Button** then click on the drop down box at the left and select **Runge Kutta 4 Auto**.
- Click on the **Simulate Button** then double click on *Angular Position* and click on the Graph tool.



This is more like what you would expect - continued oscillation to plus and minus 20 degrees. Changing the initial position to 45 degrees (no longer a small deviation) gives:



You get the same qualitative behavior, but the period is longer. This is different from what is typically learned in introductory physics, in which the period is constant due to the approximation of $\sin(\Theta) = \Theta$. You might change the model so that:

```
angular acceleration = -radian2degree *  
                      (Angular Position/radian2degree) * g / length
```

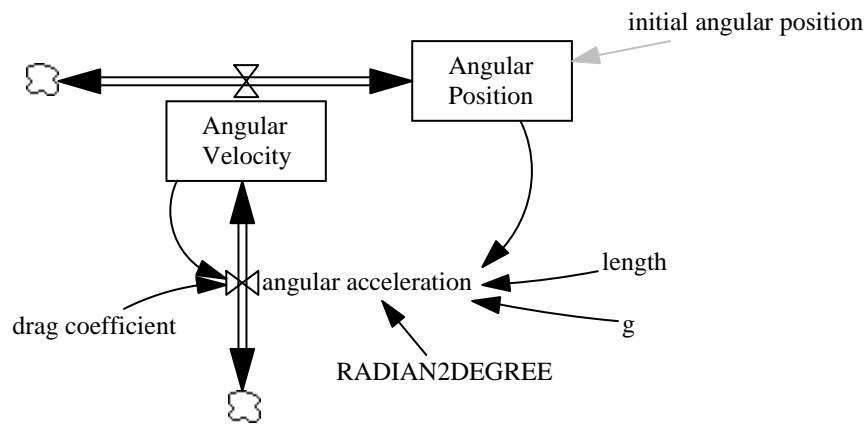
and see what behavior results.

The Damped Pendulum

If you look back at the workforce inventory model discussed in Chapter 16 you will see that it is remarkably similar to the pendulum model with *Inventory* analogous to *Angular Position* and *Workforce* analogous to *Angular Velocity*. However, if you count the number of feedback loops, either by inspection or using the Loops tool, you will notice an important difference. There is only a single feedback loop that goes through *Angular Velocity*, but there are two feedback loops that go through *Workforce*. One of these is the analog of the pendulum's feedback loop, the other, however, is a short loop that goes directly to *net hire rate* and back.

A feedback loop that involves only a single level is called a minor feedback loop. Minor negative feedback loops are important because they are pervasive, and because they tend to be stabilizing. That is, strengthening a minor negative loop will tend to make behavior more stable.⁴ If we take into account the fact that as a pendulum moves, there is drag on its motion, we need to introduce a similar negative loop. We modify the pendulum model to incorporate this change, making the simple assumption that drag is proportional to velocity.

⁴There are exceptions to this rule of thumb and that is one of the reasons why it is important to actually simulate the models you are working with.



The new equations are:

```
angular acceleration=
  -RADIANT2DEGREE*SIN(Angular Position/RADIANT2DEGREE) * g /
    length - Angular Velocity * drag coefficient
Units: Degree/Second/Second

drag coefficient=0.2
Units: 1/Second
```

If you simulate this model (be sure to use Runge-Kutta 4 integrations) you should see damped oscillations. Note that if you set *drag coefficient* to 0.0 this model will exhibit the same behavior as the previous model.

It is interesting to experiment with different values for *drag coefficient*. It is also interesting to change this while using Euler integration and compare the results to the more exact Runge Kutta simulations. For a given value of TIME STEP find out how big you need to make *drag coefficient* in order to get practically the same results from Euler and Runge-Kutta integration.

Conclusions on Integration Techniques

The dramatic differences between the results of Euler and Runge-Kutta integration are rarely as clear as they are in this chapter. For most models of social systems, the different techniques do not lead to dramatically different results. For physical systems, in which the interrelationships are exact and based on physical laws, Runge-Kutta integration is almost always preferable.

You build and use models in order to address problems. As these models are built, there are various checks that must be done against reality. These checks may be explicit and take the form of tests of model behavior or subsector behavior under different assumptions, or they may be implicit mental simulations and your own understanding of models and the modeling process. In either case these checks are very important in ensuring that the models you develop can adequately address the problems they are being applied to.

Reality Check gives you a straightforward way to make statements you think must be true about a model for it to be useful, and the machinery to automatically test a model for conformance with those statements. Reality Check is a new technology that adds significantly to your ability to validate and defend the models you build. It can also focus discussion away from specific assumptions made in models onto more solidly held beliefs about the nature of reality.

This chapter:

- Introduces the concept of a Reality Check.
- Shows you how to define Constraints and Test Inputs.
- Shows you how to test a model for conformance to Reality Check equations.
- Provides a simple example of model building using Reality Check.

Models and Reality

Models are representations of reality, or our perceptions of reality. In order to validate the usefulness of a model, it is important to determine whether things that are observed in reality also hold true in the model. This validation can be done using formal or informal methods to compare measurements and model behavior. Comparison can be done by looking at time series data, seeing if conditions correspond to qualitative descriptions, testing sensitivity of assumptions in a model, and deriving reasonable explanations for model generated behavior and behavior patterns.

Another important component in model validation is the detailed consideration of assumptions about structure. Agents should not require information that is not available to them to make decisions. There needs to be strict enforcement of causal connectedness. Material needs to be conserved.

Between the details of structure and the overwhelming richness of behavior, there is a great deal that can be said about a model that is rarely acted upon. If you were to complete the sentence "For a model or submodel to be reasonable when I _____ it should _____" you would see that there are many things that you could do to a model to find problems and build your confidence in it.

In most cases, some of the things required for a model to be reasonable are so important that they get tested. In many cases, the things are said not about a complete model but about a small component of the model, or even an equation. In such cases you, as a model builder, can draw on your experiences and the writings of others relating to behavior of generic structures and specific formulations.

Ultimately however, most of the things that need to be true for a model to be reasonable are never tested. Using traditional modeling techniques, the testing process requires cutting out sectors, driving them with different inputs, changing basic structure in selected locations, making lots of simulations, and reviewing the output. Even when this gets done, it is often done on a version of the model that is later revised, and the effect of the revisions not tested.

Reality Check equations provide you with a language for specifying what is required for a model to be reasonable, and the machinery to go in and automatically test for conformance to those requirements. The specifications you make are not tied to a version of a model. They are separate from the normal model equations and do not interfere with the normal function of the model. Pressing a button lets you see whether or not the model is in violation of the constraints that reality imposes.

Domains of Expertise

Although Reality Check equations in Vensim have been implemented as an extension of the Vensim modeling language, the skills and experience required to write Reality Check equations is different from that required to write models. Reality Check equations are assertions about the nature of behavior in reality. They do not require the creation of structure capable of generating particular behavior. Reality Check equations create behavioral conditions and then check to see if the structure of the model causes the appropriate behavioral response.

Because Reality Check equations are formulated in a behavior-behavior world, the people best at formulating them are the people with the most knowledge of behavior—for the most part experts in the domain of study. Because of this, Reality Check equations can do much more than find bugs in models. Reality Check equations allow the consumers of models to have confidence in the quality of the results they are getting.

Defining Reality Check Equations

You enter Reality Check equations in the same way you enter other equations in Vensim. You can use the Sketch Tool to define inputs to Reality Check equations, or simply write the equations directly in the Equation Editor or Text Editor. Structurally, Reality Check equations are not inputs to any model equations, but use model variables in their definitions. When Reality Check equations are exercised, they can change the value of model variables as well as the equations used to compute those variables. Again, we emphasize that Reality Check equations are not statements about causal structure but simply statements about behavior—"If this happens, then that must happen".

The appropriate naming conventions for Reality Check equations are different from those for model variables. Model variables should be named as nouns or noun phrases which more and less have obvious meaning—*Workforce*, *productivity*, *capacity*, *determination*, *propensity to save* and so on. Reality Check equations, on the other hand, should be brief phrases that describe the nature of the check—*no workers no production*, *rain means flooding* and so on. The best guide in naming Reality Check equations is to think of them as true or false, and name the Reality Check the statement that would be made when it took on a true value.

There are two types of equations that can be defined in Vensim to make use of the Reality Check functionality—Constraints and Test Inputs. Constraints make statements about the consequences that should result from a given set of conditions. They are called Constraints because they specify the way in which the Test Inputs should constrain behavior. The violation of a Constraint indicates a problem

with the model. Test Inputs are a way of specifying the conditions or circumstances under which a Constraint is binding. Since Test Inputs can be used in Constraints, they are described first.

Test Inputs

Test Inputs allow you to define alternative conditions by changing equations for a variable in the model. The basic format for a Test Input is:

```
name :TEST INPUT: variable = expr
```

Where *name* is the name of a Test Input. What appears to the right of :TEST INPUT: is exactly the same format equation you would normally use to define an auxiliary variable, and can involve only model variables. The equation you write is also restricted in that you cannot use dynamic functions (such as SMOOTH), defining functions (such as ACTIVE INITIAL) or user defined Macros. If you need this functionality, you can create extra model variables for use in your Test Inputs.

You can only use Test Inputs in the conditional portion of a Constraint equation. The major reason for defining Test Inputs is to give a name to the experiment being conducted. This can make reading the Constraint much easier. If you do not define Test Inputs, you can define Constraints directly using the *variable* = *expr* portion of the Test Input equation. The same restricted equation format applies if you do this.

Dynamic Test Inputs

In addition to an alternate assignment expression for a Test Input, it is often desirable to force a change in a variable after a period of time in a simulation. For example, you might want to force production to ramp down to 0 between time 10 and 12, but before time 10 just let production be what it would have been. This type of a change is useful both for writing complete reality Checks and for studying the response of a model to interesting Test Inputs..

To create Test Inputs with a time profile there are a series of functions that begin with RC — RC COMPARE, RC DECAY, RC GROW, RC RAMP and RC STEP (details in Chapter 9). These all behave in a similar manner. For example:

```
TI Production to zero :TEST INPUT:
    production = RC RAMP(production,0,2,10)
```

This Test Input will cause production to go to ramp from whatever it is at time 10 to zero at time 12.

The RC... functions all take two optional arguments—a start time and a duration. If the duration is omitted, as it is above, the Test Input continues in the changed state to the end of the simulation. If you specify a duration the Test Input will continue in the changed state for the time specified, and then the variable will revert to its normally computed value. When it does revert the values the variable depend upon will likely be different so it most likely will not take on the value it has in a normal simulation.

If the start time (10 in the above example) is omitted, and the model contains the constant RC START TIME the changed specified will begin at this time. If RC START TIME is not in the model the change will start at INITIAL TIME + TIME STEP. Using RC START TIME in this manner is convenient because it allows you to globally change the time at which changes take effect and allows you to leave off additional arguments to the RC functions. Having Test Inputs start during a simulation is helpful because it prevents startup behavior from interfering with the testing of the model and also permits you to run tests with different relative values for variables.

Constraints

Constraints take the form

```
name :THE CONDITION: condition :IMPLIES: consequence
```

:THE CONDITION: and :IMPLIES: are special keywords in Vensim. *condition* and *consequence* are logical expressions described below. The name of a Constraint must use letters and numbers just as other variables in Vensim. Constraints do not need units of measure, though if you are using the Text Editor you must put in the tilde symbol ~ as a placeholder. You can attach comments to Constraints just as you do with other variables in Vensim.

Logical Expressions

Constraints use a condition and a consequence that are both defined as logical expressions. An example of this would be:

```
no capital no production :THE CONDITION: Capital = 0 :IMPLIES:
production= 0
```

When testing Reality Check equations, Vensim will force a condition to be true whether the model generated values suggest it should be true or not, and test the consequence for truth. When the condition is true, but the consequence is not, Vensim reports the problem as a Reality Check failure. Vensim also does passive testing as described below.

Logical expressions can be more complicated than this. They are built up by using the comparisons =, >, <, and <> along with :OR:, :AND: and :NOT:. A valid logical expression could be:

```
Population > 8E9 :AND: (food ratio < .75 :OR:
Pollution > critical pollution)
```

Here a number of things are being compared and this expression will be true if *Population > 8E9* and either *food ratio < .75* or *Pollution > critical pollution* or both. Logical expressions can quickly become difficult to understand and you are encouraged not to combine too many things in a condition. In the consequence portion, it is often useful to have many items combined with :AND:s (to test several consequences of a single condition), but more complicated structures are rarely helpful.

The condition portion of a Constraint definition is restricted to the comparison of variables to other variables and variables to numbers. The only exception to this is that you can use *var=expr*, or a named Test Input as one of the condition's logical components. Thus:

```
pop lt cc :THE CONDITION : Population < Carrying Capacity * 1.1
:IMPLIES: deaths from crowding < 1000
```

is wrong because it takes the form *var<expr*, whereas

```
pop lt cc :THE CONDITION : Population = Carrying Capacity * 1.1
:IMPLIES: deaths from crowding < 1000
```

uses *var=expr* and is a legitimate expression. Expressions included directly in Constraint conditions in this manner can use *TIME TRANSITION* and must conform to the rules for forming Test Inputs.

Test Inputs may be used for the condition of a Constraint, as in:

```
pop lt cc :THE CONDITION : pop at cc plus 10
:IMPLIES: deaths from crowding < 1000
```

where *pop at cc plus 10* is a Test Input. Note that Test Inputs are treated as logical variables taking on a value of true if they are active and false if they are not.

All components of a Constraint are restricted to using simple functions (MIN, MAX, SUM etc.). Other than this, there is no restriction on logical expressions in the consequence portion of a Constraint definition.

It is not generally useful to test equality in a consequence because equality tests are very likely to fail even when there is nothing wrong with the model. This is because even for concepts that are definitionally equivalent, but computed in different manners, there are likely to be slight numerical differences which will be flagged during an equality test.

Dynamics Tests in the Consequences

Reality Check equations that have Test Inputs using RC... functions typically use a corresponding RC...CHECK function in the consequence. The RC ... CHECK functions work in an analogous way to RC... functions in Test Inputs. While Test Inputs change the value of a variable, the consequence portion of a Constraint makes a comparison of the value of a variable. The RC...CHECK function takes one more argument than the corresponding RC... function. This argument is the grace period and it allows a delay after a Test Input occurs before the consequences are tested. For example:

```
TI Production to zero :TEST INPUT:
    production = RC STEP(production,0)
RC No Production no Shipments :THE CONDITION:
    TI Production to zero :IMPLIES:
        sales <= RC RAMP CHECK(.5,sales,.0001)
```

The Test Input causes *production* to step to zero at RC START TIME. After a grace period of .5 sales is checked to see if it is less than or equal to .0001 times the value it had at RC START TIME. The use of .0001 instead of 0 prevents nuisance violations that might occur with continuous formulations and is good practice.

The grace period is the first argument to all the RC...CHECK functions except for RC COMPARE CHECK which takes uses the literal name of a file first.

:CROSSING:

In the implication portion of a Reality Check you can use :CROSSING: and :AT LEAST ONCE: with > and < to look for above/below type relations, as in:

```
... :IMPLIES:
    Inventory > :CROSSING: RC STEP CHECK(0,Inventory,1)
```

This will test to be sure that at RC START TIME Inventory is first bigger than its baseline value and then becomes smaller and stays smaller. If there were two :CROSSING: keywords in a row as in:

```
Inventory > :CROSSING: :CROSSING: RC STEP CHECK(0,Inventory,1)
```

Inventory would need to start bigger, then become smaller, then become bigger and stay bigger.

If you use one or more `:CROSSING:` keywords, you can end the sequence with `:IGNORE:` to indicate that after the required number of crossings have been made, it does not matter if any more crossings occur. For example

```
Inventory > :CROSSING: :IGNORE: RC STEP CHECK(0,Inventory,1)
```

This says that *Inventory* needs to start bigger, then become smaller, then it does not matter what it does.

:AT LEAST ONCE:

Analogous to the `:CROSSING:` keyword is a `:AT LEAST ONCE:` keyword that simply requires that the condition be true once after RC START TIME. For example

```
Inventory > :AT LEAST ONCE: RC STEP CHECK(0,Inventory,1)
```

says that *Inventory* needs to exceed its value at RC START TIME at least once during the rest of the simulation. It might always be above, or cross from below to above. If the value is above once it can also cross below and the condition will remain true.

Empty Conditions

The condition part of a Constraint may be empty as in:

```
debt bounded :THE CONDITION: :IMPLIES: debt < 4E6
```

Which states that no matter what happens there will never be more than four million in debt. Note that Vensim does not try to test all possible model conditions when it sees an empty condition. Constraints with empty conditions are checked passively anytime you are using the Reality Check function, and will detect a high debt.

For the simple example given here, using 4E6 as the maximum value that debt could take on in the equation for debt would also result in a message when debt exceeds 4E6 as long as warnings are not suppressed. The Empty condition can, however, be used to evaluate much more general expressions.

Wildcard Tests in the Consequence

In addition to testing a variable, you can test all variables to see if they satisfy a condition. To do this use a `*` instead of a variable name. For example you might write the Constraint

```
all peaceful :THE CONDITION: FINAL TIME=101 :IMPLIES:  
* < 1E9 :AND: * >= -1E3
```

which will test to be sure that all variables stay in the range -1,000 to 1 billion. The condition *FINAL TIME = 101*, which simply runs the simulation an extra year, is used instead of the empty condition because this is a time consuming check to make and Constraints with empty conditions are checked passively every time any Reality Check is being made.

Simulation and Reality Check

Before we discuss the mechanics of actually running Reality Check equations, it is useful to describe very briefly what happens inside the model. Reality Check equations involve systematic intervention in the basic structure of the model. They are qualitatively different from sensitivity analysis in that there are not any well defined pathways of influence. Test Inputs and Constraints can cause changes to be made at almost any point in a model.

In order to accomplish the changes involved in running Reality Check equations, Vensim restructures your model, adding equations and modifying the sequence in which equations are computed to match. After completing Reality Check equations, Vensim returns the model to its original structure. This means that doing causal tracing on a run made with one or more Test Inputs active can give surprising and seemingly incorrect results.

In some cases, restructuring and reordering may leave the model ill formed. The most common problem is that the model may contain simultaneous equations and therefore not be computable. If this is the case, Vensim will report the problem and not complete the simulation. Because of the behavior-behavior nature of Reality Check equations, the existence of simultaneous equations does not necessarily represent a conceptual problem, but will prevent simulation. If possible, reformulate the Test Inputs causing the problems to avoid simultaneous equations.

NOTE Reality Check equations are always run using interpreted (not compiled) simulations. This is because they require continual restructuring of equations.

Active Constraint Checking

During active Constraint testing, Vensim forces the condition part of Constraint equations to be true, changing variable values or model structure where necessary.

- If the condition is an equality condition or Test Input, Vensim appends the equation to the existing model equation (remember that the equation may reference the original computed value for a variable). The new value for the variable is then used wherever the variable is used.
- If the condition uses inequality, Vensim first tests to see if the inequality is true.
 - If it is true, the value of the variable is not changed.
 - If it is not true, Vensim makes it true by assigning the value just as if it were an equality condition.

Having forced conditions to be true Vensim tests to see if the consequences are also true.

Passive Constraint Checking

In passive Constraint checking, Vensim simply evaluates both halves of the Constraint equations as logical expressions. If a consequence is false while its condition is true, an error is reported.

Whenever you run a Reality Check, Vensim tests all Constraints that are not explicitly activated for passive conformance. This checking does not get done during normal simulation.

Error Reporting

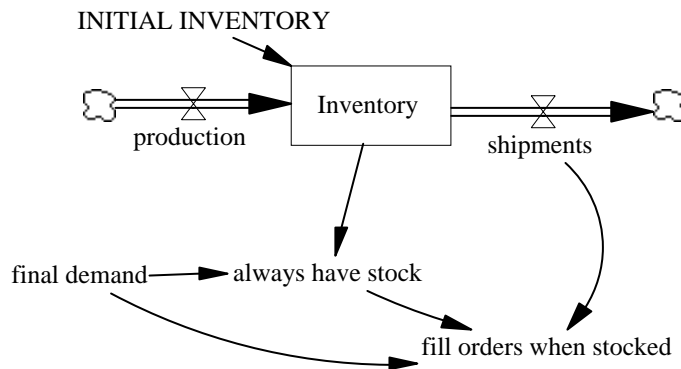
Violations of Constraints are reported the same way Lookup table overflows are. The first time a Constraint is violated an error is reported. Then a message is sent when the Constraint is no longer violated indicating a return to conformance. The next violation is reported, and so on.

Entering Reality Check Equations

You enter Reality Check equations the same way you enter normal model equations. In the Text Editor, you simply type them in. In the Sketch Editor, you can enter them in diagrams by showing all of the elements to be checked as causes of the Test Inputs and Constraints. The Test Inputs and Constraints are not part of the model's causal structure. In addition, the right hand side of alternative defining equations in Test Inputs are not shown as causes of the variable being given the Test Input. Thus the equations:

```
Inventory = INTEG(production-shipments,INITIAL_INVENTORY) ~~|
always have stock :TEST INPUT: Inventory = 3*final demand ~~|
fill orders when stocked :THE CONDITION: always have stock
:IMPLIES: shipments >= final demand ~~|
```

would appear on the diagram as:



In it is likely that you will want to keep the equations and causal structure used in defining Reality Check equations separate from the normal model equations. One convenient way to do this is to put them in a separate group and include them on separate views.

In the diagram, the mixing of the different types of names can be confusing. When looking for arrows going in to *fill orders when stocked*, it is not what causes *fill orders when stocked* that is wanted, but what we need to know to determine if this is true. Given the difficulties of understanding diagrams just showing causal connections and flows, you may want to omit this added complexity from the working diagram. It is probably easiest to place Reality Check equations in a separate view so they will not be confused with model structure.

It can also be very convenient to adopt a naming convention for Reality Check statements. For example you might start all Test Inputs with TI and all Constraints with RC.

Equation Editor


You enter Reality Check equations just like you would enter other equations. Create a variable, open with the Equation Editor, select the **Type** Reality Check and the **Suptype** Constraint or Test Input and fill in the equation.

The screenshot shows the 'Equation Editor' dialog box with the title 'Editing equation for - hunger from growth'. The main text area contains the equation: `Sugar <= RC DECAY CHECK(1,Sugar,0.5,INITIAL TIME)`. Below the equation, the 'Type' dropdown is set to 'Reality Check' and the 'Constraint' dropdown is set to 'Constraint'. There is an unchecked checkbox for 'Supplementary' and a 'Help' button. To the right of the equation, there is a 'Variables' tab, a 'Functions' tab, and a 'More' tab. Below these tabs is a 'Choose Initial Variable...' button and a list of variables: 'big growth', 'INITIAL TIME', and 'Sugar'. At the bottom, there is an 'Errors' field showing 'Equation OK' and several buttons: 'OK', 'Check Syntax', 'Check Model', 'Delete Variable', and 'Cancel'.

When you select **Type** Reality Check, you will notice that the **MinMax** label changes to **TypPrio** (this appears near the bottom just to the right of **Group**). This is a mechanism for filtering Reality Check statements in the Reality Check Control dialog and will be discussed below.. You can enter a number for both Type and Priority. The Values that Priority can take on are arbitrary, but 1-10 would be a common prioritization. Type should take on an integer value between 0 and 64.

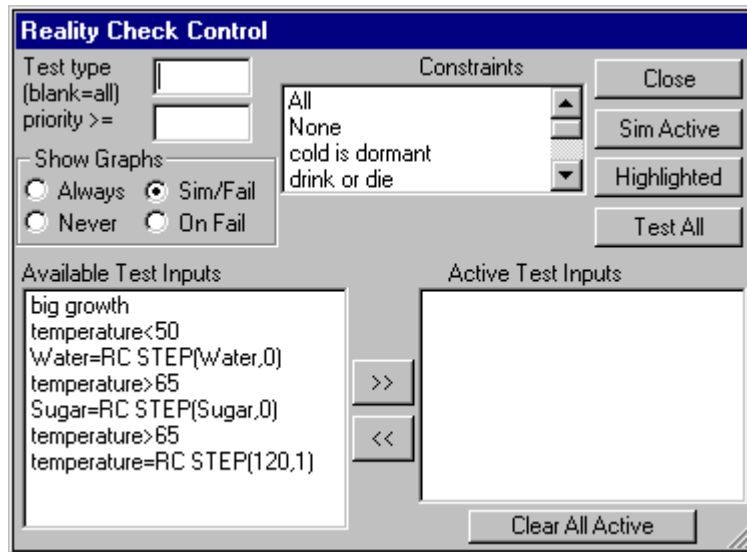
Running Reality Check

Reality Check is run by first setting up a simulation and then testing one or more of the Reality Check equations you have entered.

You start Reality Check from the Toolbar . You can make changes and adjust other options just as you would for a normal simulation. After you have done this click on the **Reality Check** button  on the Toolbar.

NOTE Any changes you have made to constant values, data to be used, and so on prior to launching Reality Check will continue to hold throughout the Reality Check session.

When you launch Reality Check, the Reality Check Control dialog appears:



Test type allows you to specify which type of Reality Check you want to test. If this is blank all types will be tested. The type of a Reality Check is indicated in the **Typ** field of the Equation Editor. In the Text Editor Type, and Priority are enclosed in square brackets [] in the units field (~[type,priority]). This field is only applicable when the **Test All** button is used to start testing.

Priority >= will restrict testing to only Constraints having a priority bigger to or equal to that specified. This field is only applicable when the **Test All** button is used to start testing.

NOTE If Constraints do not have a priority specified, they are treated as highest priority. If Constraints do not have a type specified they are assumed to match all types.

Show Graphs is used to display a graph of the variable being tested in the Consequence portion of a Constraint against the behavior that variable needs to conform to. This is done using a special invocation of the Graph Tool and can be very helpful for understanding the manner in which the failure occurred.

- **Always**, if checked, causes a graph to come up for each Constraint Checked.
- **Never**, if checked, suppresses graph output.
- **Sim/Fail**, if checked, causes graphs to appear whenever a Reality Check fails, and also when a single simulation is made using the **Sim Active** or **Highlighted** buttons.
- **On Fail**, if checked, causes a graph to come up only if a Reality Check fails. When you use the **Test All** button this is the same as **Sim/Fail**. For the **Sim Active** and **Highlighted** buttons this suppresses the graph unless there is actually a failure.

Constraints is a list of Constraints you have entered. Clicking on one of these will highlight the corresponding Test Inputs. You can then activate one or more of these Test Inputs.

Test Inputs is a list of the Test Inputs in the model. This list includes everything you defined explicitly as a Test Input, and all comparisons in the logical expressions making up the condition components of Constraint equations. Comparisons are shown directly and not given a name.

>> moves the highlighted items in the Test Inputs list to the Active Test Inputs list.

<< removes the highlighted items in the Active Test Inputs list.

Clicking on an element in the list highlights it. Control-clicking toggles the highlight status adding or removing it from the selection of highlighted elements. Double clicking moves the item to the Active Test Inputs list.

Active Test Inputs shows a list of the Test Inputs, explicit and implicit, that are active. These will be used when the Simulate Button is clicked. Clicking on an element highlights it, and unhighlights anything else that may be highlighted. Control-clicking toggles an element's highlight. Double clicking removes an element from the list.

Clear All Active removes all elements of the Active Test Inputs list.

Sim Active simulates the model using the active Test Inputs in the list. All other Constraints are tested passively. If the simulation succeeds the results will be stored just as for a normal simulation, and you can review them with the tools on the Workbench. Be careful to note that causal tracing will not always give the expected results since model structure may have been changed during the simulation.

Highlighted simulates the model using the item highlighted in the Constraint list. Because of the logical structure of the condition portion of a Constraint this may actually require more than one simulation. The final simulation will be stored just as a normal simulation.

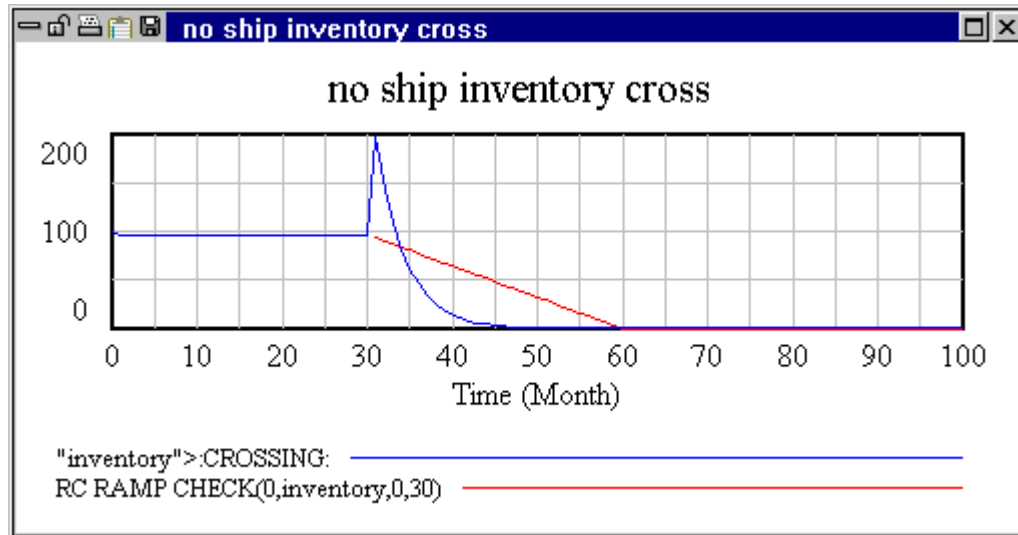
Test All tests all the Constraints in the model one at a time by forming the collection of Test Inputs necessary to activate each Constraint. Because of the logical structure of the condition portion of a Constraint equation, it may take more than one simulation to test a Constraint. This testing can be time consuming, activity and errors are reported in a separate window. Running Test All does not store any simulation results but simply reports the results of Reality Check.

Close closes the Reality Check Control dialog.

Reality Check Results

Reality Check results are reported in a text window. The window shows which Constraints were checked and whether any Constraint was violated. A new window is brought up each time you click on **Sim Active**, **Highlighted** or **Test All**. Examples are given below.

One or more graphs may also appear. These graphs are intended to display the implication portion of a Constraint:



Here the spiked line shows what inventory does while the sloped line shows what it is being compared to.

Reviewing Simulation Results

Each time you click on **Sim Active** or **Highlighted** a new simulation is made. That simulation is given the name currently specified in the Runname box (on the Toolbar or in the Simulation Control dialog). You can, while the Reality Check Control is open, look at the results in the simulation just as you would any other simulation. You can then make another simulation and review those results. Each simulation you make will overwrite the last one. If you want to compare two Reality Check simulations, you should close the Reality Check Control and then start again placing a different name in the simulation Runname box.

NOTE If you change constants in the Simulation Control dialog or on screen using Set Up a Simulation from the Toolbar, those constant changes will be held throughout your Reality Check session.

Reality Check and Yeast Growth

To demonstrate the mechanics of writing and using Reality Check equations, it is useful to work through a very simple example. Suppose that you want to model the growth of yeast in a bowl of water. The water is held at a constant temperature that can be set, and has a fixed amount of sugar in it to start.

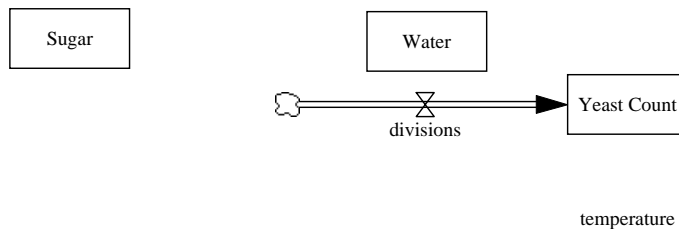
We first list some of the Reality Check equations that must be satisfied:

- If temperature is below 50 degrees yeast growth should stop, and the yeast go dormant.
- If temperature is above 100 degrees the yeast should die out.
- If there is no sugar and the yeast are not dormant the yeast should die out.
- If there is no water and the yeast are not dormant the yeast should die out.
- If the yeast continue to grow they will consume all the sugar.
- If the yeast continue to grow they will consume all the water.

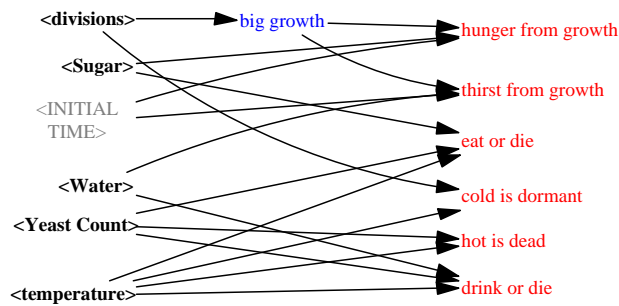
The other thing we know is that yeast reproduce by dividing and that when conditions are right they can reproduce with a doubling time of about 10 minutes.

Test Input and Constraint Equations

We will start the modeling process by defining the variables that we need to test our Reality Check equations and then entering our Reality Check equations. The Reality Check equations we have outlined talk about the number of yeast, the amount of water, the amount of sugar, the number of yeast divisions and temperature. Identifying levels, we put these into a skeletal model as:



For clarity, we want to build up our Reality Check equations in a second view. In that view we will place all of the elements above as shadow variables, and use the first view to evolve the model and the second view to manage the Reality Check equations.



There is no fixed set of rules for structuring Reality Check diagrams. In our experience, making a column for normal model variables, Test Inputs and Constraints is the easiest thing to follow. It can also be helpful to color code (e.g., blue for Test Inputs and red for Constraints). The arrows can get pretty messy this way, but if you organize Reality Check information by the variables that are affected, this is usually not a big problem.

You can enter the Constraints and Test Inputs in the Equation Editor:

You will need to select the equation **Type** Reality Check and the subtype Constraint. The conditional and consequence are split into two separate windows.

The Reality Check equations are:

```
cold is dormant :THE CONDITION: temperature < 50 :IMPLIES:
    divisions = 0
```

```
hot is dead :THE CONDITION: temperature = RC STEP(120,1)
    :IMPLIES: Yeast Count <= RC DECAY CHECK(1,Yeast Count,2)
```

For *hot is dead* we have used a step in the *temperature* and then used a RC DECAY CHECK function to check *Yeast Count*. For this model a decay is appropriate for looking at the behavior of *Yeast Count* since we are focusing in on situations of dying out.

```
eat or die :THE CONDITION: Sugar = RC STEP(Sugar,0)
:AND: temperature > 65 :IMPLIES:
    Yeast Count <= RC DECAY CHECK(1,Yeast Count,15)
```

Here, the Constraint requires the *Yeast Count* decline toward zero with an average death time of 15.

```
drink or die :THE CONDITION: Water = RC STEP(Water,0)
:AND: temperature > 65
:IMPLIES: Yeast Count <= RC DECAY CHECK(1,Yeast Count,1)
```

```
big growth :TEST INPUT: divisions = 1e+022
```

```
hunger from growth :THE CONDITION: big growth :IMPLIES:
    Sugar <= RC DECAY CHECK(1,Sugar,0.5,INITIAL TIME)
```

```
thirst from growth :THE CONDITION: big growth :IMPLIES:
    Water <= RC DECAY CHECK(1,Water,0.5,INITIAL TIME)
```

Notice that for the last two Constraints the RC DECAY CHECK function starts checking at INITIAL TIME since the Test Input starts from the beginning of the simulation. If the Test Input had used an RC STEP function to start during the simulation the INITIAL TIME argument would have been left off of the last two RC DECAY CHECK uses.

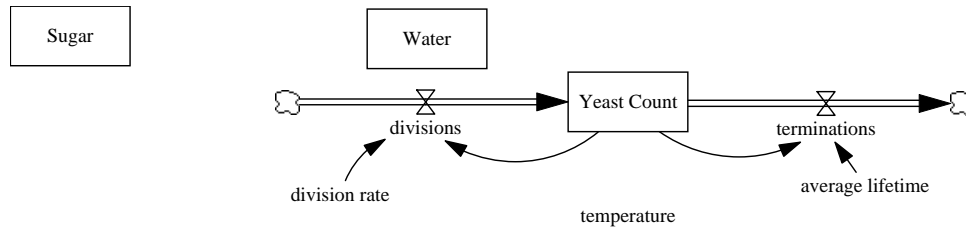
Finally we have:

```
RC START TIME = 10
```

The time is arbitrary. It might be that starting early there will be few yeast and lots of sugar, while starting later there would be more yeast and less sugar. When a model passes Reality Check equations to your satisfaction it is a good idea to change RC START TIME and retest.

An Initial Model

We have a short list of things we know will need to be in our model just to make it possible to use Reality Check equations. Armed with a very basic understanding of exponential growth, we can fill in our skeleton framework to build a model. Suppose we start with the simplest possible model (*yeast1.mdl*):



INITIAL TIME = 0, *FINAL TIME* = 300, *Units: Minute*

average lifetime = 250

Units: Minute

division rate = 0.08

Units: 1/Minute

divisions = *Yeast Count***division rate*

Units: Cell/Minute

Sugar = 100

Units: g

temperature = 85

Units: Farenheit

terminations = *Yeast Count*/*average lifetime*

Units: Cell/Minute

TIME STEP = 1

Units: Minute

Water = 100

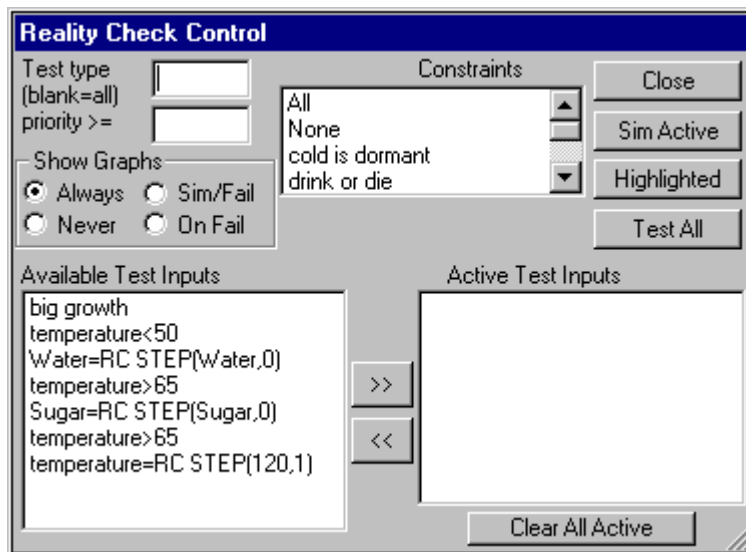
Units: ml

Yeast Count = *INTEG*(*divisions-terminations*,100)

Units: Cell

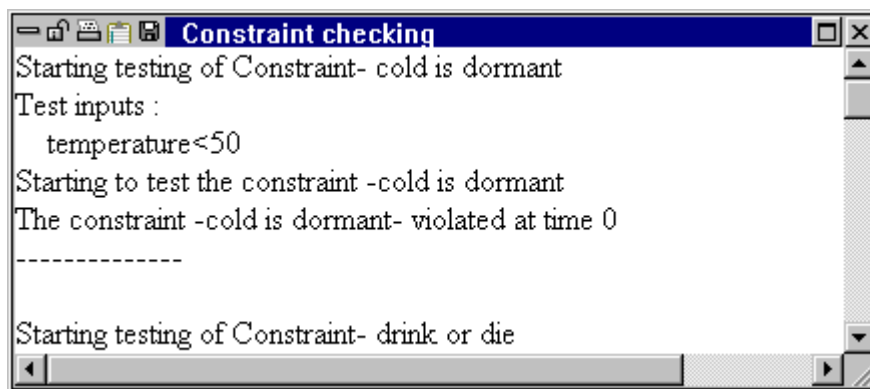
In this model we are not making use of *Sugar*, *Water* or *Temperature*, and they have been set to constants with the **Type** drop down box in the Equation Editor.

Next, we run our Reality Check equations on this model. Click on the **Reality Check** button on the Toolbar, or open the Simulation Control Dialog and click on the **Reality Check** button.



In the Reality Check Control dialog you will see a list of all the Constraints for the model. Below that is a list of Test Inputs. One, *big growth*, was named explicitly while the others are simply derived from the conditional parts of the different Constraints.

Clicking on Test All begins a series of simulations. The following violations are reported:



The condensed report is:

18: Vensim PLE User's Guide

Starting testing of Constraint- cold is dormant

Test inputs :

temperature<50

Starting to test the constraint -cold is dormant

The constraint -cold is dormant- violated at time 0

The constraint -drink or die- violated at time 11

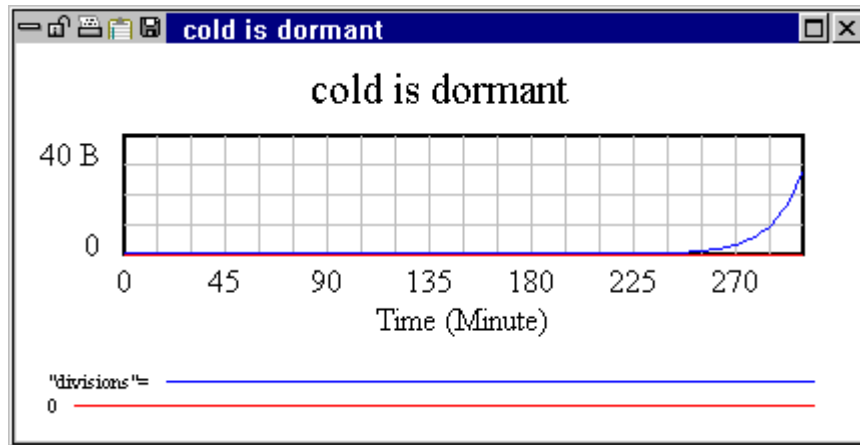
The constraint -eat or die- violated at time 11

The constraint -hot is dead- violated at time 11

The constraint -hunger from growth- violated at time 1

The constraint -thirst from growth- violated at time 1

Every Constraint has been violated. This is not a great surprise since the elements involved in the Constraints were not connected in the model. The model as presented represents a basic growth mechanism. No attention has been given to either control or containment which is what all the Constraints relate to. In addition to the error window there will be 6 graphs such as:



The top line shows what *divisions* does, while the bottom line shows what it ought to do.

The report window ends with a summary of what has happened.

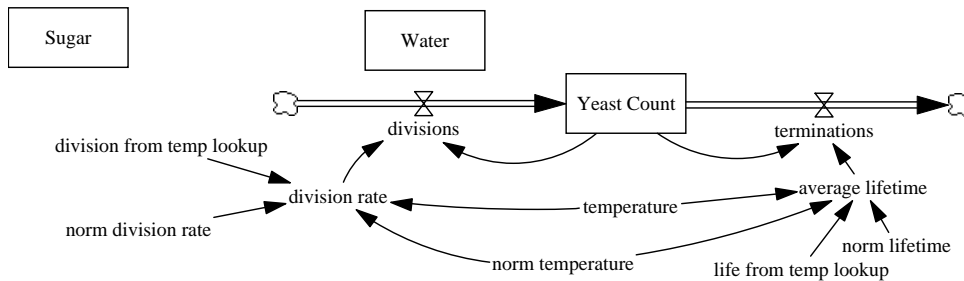
0 successes and 6 failures testing 6 Reality Check equations
 The Reality Check Index as run is 0
 Closeness score is 0.0% on 6 measurements

The first line is a summary count. The second line reports the Reality Check Index. This is defined as the number of successes divided by the product of the number of dynamic variables in the model with the total number of variables in the model. Since for every variable pair there is the potential for one or more Reality Check equations this index is something that should be near one for a model with a complete Reality Check set.

Finally a closeness score is reported. The closeness score is the average closeness of a Reality Check. If a Reality Check Passes, its closeness is 1. If it fails its closeness is 1 minus the average absolute error divided by the amount of variation in the variable being checked. Thus a Reality Check that only just fails has a Closeness score close to 1, so that the closeness score is a continuous measure of how badly, on average, Constraints have been violated.

Temperature, Divisions and Terminations

Two conditions relate temperature to growth. If temperature is low everything goes dormant. If temperature is high the yeast die. We replace the simple equations for *division* and *termination* with this in mind we create *yeast2.mdl* as:



```
divisions = Yeast Count * division rate
division rate = norm division rate * division from temp lookup (
  temperature / norm temperature )
norm division rate = 0.08
norm temperature = 80
division from temp lookup ( (0,0),(0.8,0),(1,1),(2,1) )
terminations = Yeast Count / average lifetime
average lifetime = norm lifetime * life from temp lookup (
  temperature / norm temperature )
norm lifetime = 250
life from temp lookup ((0,1),(1,1),(1.25,0.02),(2,0.001) )
```

Now testing the Constraints shows that *cold is dormant* is respected.

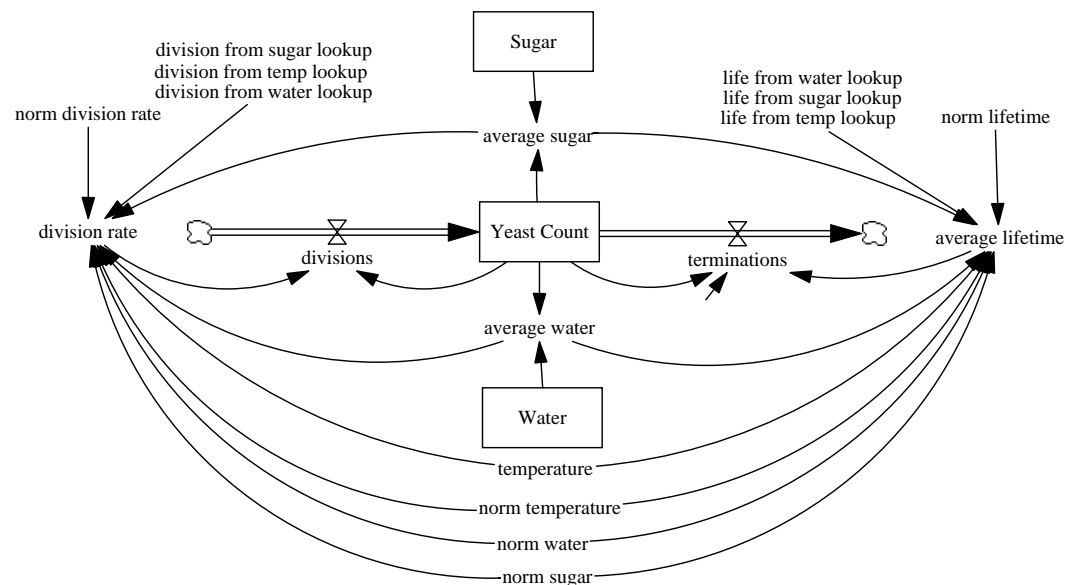
Constraint is violated because the yeast do not die out quickly enough when the temperature is increased. In this case, we can restructure the model to meet the Constraint, or relax the Constraint slightly, depending on which is more realistic. One way to relax the Constraint is to change the time transition to allow more time for the yeast to die out (we change the *RC DECAY CHECK* argument from 2 to 5):

```
hot is dead :THE CONDITION:  temperature = RC STEP(120,1)
                        :IMPLIES:  Yeast Count <= RC DECAY CHECK(1,Yeast Count,5)
```

Now testing the Constraints shows that *cold is dormant* and *hot is dead* are both respected. This kind of interplay, trading off strictness of Constraints and making adjustments to equations is valid. It provides a means of focusing more attention on the Constraints that may be violated to a small extent.

Divisions as influenced by Water and Sugar

The next two Constraint equations state that water and sugar are necessary for survival. We model the impact of insufficient water and sugar on both the division rate of the yeast and the average lifetime of the Yeast. The new model (*yeast3.mdl*) looks like:



The equations are available with the sample model. Note that the Test Inputs chosen cause extreme conditions to exist for Water and Sugar, and the way the model is formulated makes *terminations* skyrocket. Using Euler integration, the system becomes a 1 period oscillator with Yeast Count going negative. To prevent this, another integration technique can be used, or the equation for *terminations* changed to:

```
terminations = MIN(Yeast Count/TIME STEP,
                  Yeast Count/average lifetime)
```

This formulation prevents uninteresting dynamics, but also can set *Yeast Count* to precisely zero. This means that *average sugar* and *average water* need to be computed as

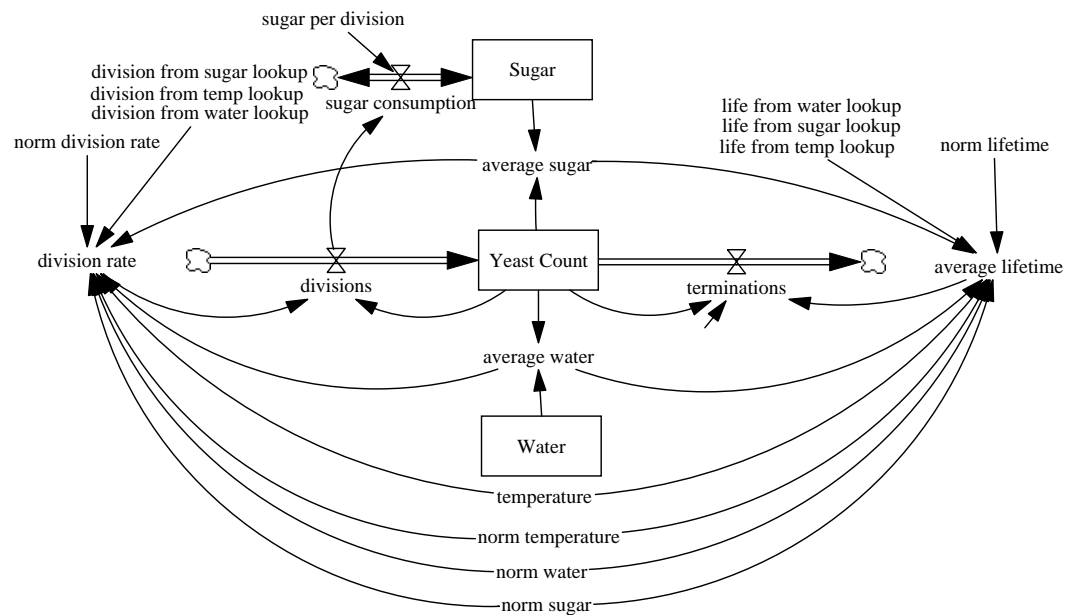
```
average sugar = ZIDZ(Sugar, Yeast Count)
average water = ZIDZ(Water, Yeast Count)
```

to prevent a numerical error (overflow) when *Yeast Count* is 0.

With the changed structure, the model now passes four out of the six Constraints.

Water and Sugar as Influenced by Divisions

Nearing the end, we still have not made the model conform to the last two Constraints. The problem is that we have not yet made a connection from what the yeast are doing to how much water and sugar there is. The simplest way to model this is to consider the division process as a consumer of sugar and water resources. Thus we have the model *yeast4.mdl*:



And this model conforms to all the Constraints written. But is it a good model? In this case the answer is probably "not yet". For a detailed and accurate representation of yeast growth we would need to get some experimental data, and calibrate the model for temperature and resource deprivation effects. The point about this model is that it does not violate the most obvious common sense Reality Check equations.

We have shown a number of different models. What about the Reality Check equations? Except for the small change made to *hot is dead*, they are unchanged and have exactly the same diagram.

Conclusion

The use of Reality Check serves two purposes. First, it is written record of things that were assumed must be true for a model to make sense. These are things that typically go undocumented, but are often the most important contribution of a model building exercise. It is often the insight that if the yeast keep growing they will run out of water that is the most important thing to learn about a system. Something that seems reasonable when said, but has far reaching consequences.

The second important thing about this example is that it illustrates a way of building models that is both effective and relatively efficient. You start from a basic core and then add the complementary structure necessary to get conformance to Constraints. This provides a rigor and direction that can increase greatly both the speed and quality of modeling work.

19 Functions and Keywords

A variety of functions are available in the Vensim modeling language. This chapter:

- Contains a full description of all the predefined functions and keywords in Vensim PLE and PLE Plus.
- Explains the syntax and usage of each function.
- Shows how each function treats units of measure.
- Discusses Lookup functions, with which you can specify an arbitrary nonlinear relationship.

The functions in this chapter are listed in alphabetical order.

There are a number of functions which define what a variable is. Defining functions must appear first on the right hand side of an equation. The most important defining function is the INTEG function, which defines a variable as a Level variable. Defining function are accessed in the Equation Editor by selecting the type and subtype for the variable.

Keywords start and end with a colon :. Key words relating to Reality Check are discussed in Chapter 18. Others appear only in the summary list.

Functions marked with a dagger † are available in Vensim PLE Plus but not in Vensim PLE.

Summary List of Functions

Table 4-1 is a summary table of the predefined functions, which you can use as a quick reference tool.

Table 4-1. Summary of Predefined Functions and Keywords.

A FUNCTION OF(#,A,B,C,,)	Shows "generic" relationships -automatically generated by Sketch tool.
ABS(A)	Returns the absolute value of A.
ACTIVE INITIAL(A,N)	Defines an Auxiliary with distinct active and initial values.
:AND:	Logical AND used with IF THEN ELSE.
:AT LEAST ONCE:	Relation must be true at least once for a Reality Check.
:CROSSING:	Lines must cross for Reality Check Consequence.
DELAY FIXED(I,T,N)	Delays the input I for a fixed time T starting with N.
DELAY1(I,T)	First order exponential delay of I for time T conserving I.
DELAY1I(I,T,N)	Same as DELAY1 but starting at N.
DELAY3(I,T)	A third order exponential delay of I for time T conserving I.
DELAY3I(I,T,N)	Same as DELAY3 but starting at N.
EXP(X)	Returns e(2.718...) to the X power.
GAME(X)†	Returns X except during gaming mode the user input is used.
GET 123 CONSTANTS('f','t','c')†	Returns constants from tab 't' of the 123 file 'f' starting at cell 'c'.
GET 123 DATA('f','t','r','c')†	Returns data from tab 't' of the 123 file 'f' starting at cell 'c' time in row or column 'r' (if 'r' is a number time is read across, if a letter time is read down).
GET XLS CONSTANTS'f','t','c')†	Returns constants from tab 't' of the 123 file 'f' starting at cell 'c'.

19: Vensim PLE User's Guide

GET XLS DATA('f','t','r','c')†	Returns data from tab 't' of the 123 file 'f' starting at cell 'c' time in row or column 'r' (if 'r' is a number time is read across, if a letter time is read down). Returns X if condition is non-zero, otherwise Y.
IF THEN ELSE(cond,X,Y)	Used in Reality Check.
:IGNORE:	The implication portion of a Reality Check.
:IMPLIES:	Returns the value of A at initialization time and holds it constant.
INITIAL(A)	Performs numerical integration of R starting at N (defines a Level).
INTEG(R,N)	Returns the integer part of X.
INTEGER(X)	Returns the natural logarithm of X.
LN(X)	Returns the larger of A and B.
MAX(A,B)	Returns the smaller of A and B.
MIN(A,B)	Returns the remainder resulting from dividing X by B.
MODULO(X,B)	Special symbol denoting the missing value used instead of a number.
:NA:	Logical NOT used with IF THEN ELSE.
:NOT:	Logical OR used with IF THEN ELSE.
:OR:	A pulse of height 1.0, starting at time A and lasting B time units.
PULSE(A,B)	0 until T1 then a ramp with slope S until T2 and then constant.
RAMP(S,T1,T2)	Returns a normally distributed random number between m and x with mean h and standard deviation r using noise seed s.
RANDOM NORMAL(m,x,h,r,s)	Returns a uniformly distributed random number between between m and x using noise seed s.
RANDOM UNIFORM(m,x,s)	Set output to f times var's value in run 'r' with optional start time and duration
RC COMPARE('r',var,f[,s[,d]])	After grace period g compare to f times var in 'r'.
RC COMPARE CHECK('r',var,g,f[,s[,d]])	Set output to e decaying over t with optional start time and duration.
RC DECAY(e,t[,s[,d]])	After grace period g compare to e decaying over time t.
RC DECAY CHECK(g,e,t[,s[,d]])	Set output to e growing at rate g with optional start time and duration.
RC GROW(e,g[,s[,d]])	After grace period g compare to e growing at rate g.
RC GROW CHECK(g,e,t[,s[,d]])	Set output to ramp to f times e over t with optional start time and duration.
RC RAMP(e,f[,st])	After grace period g compare to a ramp to e times f over ramp time t.
RC RAMP CHECK(g,e,f[,st])	Set output to f times e with optional start time and duration.
RC STEP(e,f[,st])	After grace period g compare to e times f.
RC STEP CHECK(g,e,f[,st])	Returns the sine of X measured in radians.
SIN(X)	Returns a first order exponential smooth of X over time T.
SMOOTH(X,T)	Returns a third order exponential smooth of X over time T.
SMOOTH3(X,T)	Returns a third order exponential smooth of X over time T starting at N.
SMOOTH3I(X,T,N)	Returns a first order exponential delay of X over time T starting at N.
SMOOTHI(X,T,N)	Returns the square root of X.
SQRT(X)	Returns 0 till time T and then H.
STEP(H,T)	Defines a test input equation for use with Reality Check.
:TEST INPUT:	Precedes the predicate condition of a Reality Check.
:THE CONDITION:	Returns the y value of the xy pairs L# corresponding to x.
WITH LOOKUP(x,(L#))	Returns X if dividing by zero (B =0) otherwise returns A divided by B.
XIDZ(A,B,X)	Zero (0.0) if dividing by zero (B=0) otherwise returns A divided by B.
ZIDZ(A,B)	

Detailed Function Descriptions

Functions marked with a dagger [†] are only available in PLE Plus.

A FUNCTION OF (#,A,B,C,...)

A "Generic" causal FUNCTION

NOTE This function is created automatically by the Sketch tool, is not intended for use in writing equations, and precludes simulation.

A FUNCTION OF indicates only that variables are related. It does not describe the form of the relationship. (As a result it cannot be calculated.) In the Text Editor, an equation using the A FUNCTION OF function may be followed by one or more equations containing syntax errors or incomplete causal lists. This function does not appear and cannot be used in the Equation Editor.

Unlike all other functions, A FUNCTION OF has an indeterminate number of arguments.

If you are using placeholder values, the first argument for A FUNCTION OF may be a number. This value will be displayed within curly brackets in the Equation Editor.

NOTE No units checking occurs on this function since it is intended to represent arbitrary functional relationships.

ABS(x)

ABSolute value [variable]

Returns the absolute value of X.

Same as IF THEN ELSE ($X < 0$, - X, X).

Units: ABS(any unit) --> same units

Examples

ABS (5.0) is equal to 5.0.

ABS (-5.0) is equal to 5.0.

ACTIVE INITIAL(active eq, initial eq)

Distinct ACTIVE and INITIAL Equations

Returns the active equation during simulation, except when needed for determining initial conditions, when the initial equation is returned. Normally this function is used to break a loop of simultaneous initial value equations.

NOTE In the Equation Editor the ACTIVE INITIAL function is automatically entered when you select the variable type Auxiliary and the subtype with Initial.

Restrictions: Must appear first on the right of the = sign. It defines a variable as an Auxiliary variable with a separate initialization condition.

Units: ACTIVE INITIAL(input units,input units) -> same units

Example

```
Capacity = Integ(capacity adjust, target capacity) ~~|
target capacity = Capacity * adjust from utilization ~~|
```

This would simulate properly, except that the initial conditions cannot be computed correctly: the initial conditions of *Capacity* require a value for *target capacity*, which in turn requires a value for *Capacity*. Since Vensim does not support the implied simultaneous computation, you need to use the following equation for *target capacity*:

```
target capacity = ACTIVE INITIAL(Capacity * adjust from utilization,
100) ~~|
```

This will cause *Capacity* to be initialized at 100; then the first value of *target capacity* will be *Capacity * adjust from utilization*. In general, this will *not* be 100; the initial expression is used only to compute the initial conditions of State variables.

DELAY FIXED (input, delay time, initial value)

FIXED period DELAY

Returns the value of the input delayed by the delay time. The initial value is the value of the variable on the left-hand side of the equation at the start of the simulation. The delay time can be an expression, but only its initial value is used.

Restrictions: DELAY FIXED must directly follow the equal sign. Vensim treats the variable on the left-hand side of the equation as a Level variable. In the Equation Editor choose type Level, subtype Fixed Delay.

Units: DELAY FIXED (unit , time, unit) --> unit

The input, initial value and output must all have the same units. The delay time must have the same units as TIME STEP.

Examples

```
DI = DELAY FIXED(I, 22, I)
DM = DELAY FIXED( MAX( A, B ), C, A)
```

Invalid

```
D = A + DELAY FIXED( R, 3.2, 0.0 )
D = DELAY FIXED( B, T, B) + 1
```

DELAY FIXED must follow the equal sign, and it cannot be part of a more complex mathematical expression. These formulations can be made valid by defining an auxiliary variable to perform the indicated operations.

NOTES The minimum delay time is TIME STEP and shorter delay times will have the same effect as a delay time of TIME STEP. DELAY FIXED and the other DELAY functions are discrete event functions and do not change except at TIME STEP intervals regardless of the integration technique used.

DELAY1(input,delay time)
DELAY1I(input,delay time, initial value)

exponential DELAY
exponential DELAY with Initial

Returns a exponential delay of the input. Equivalent to the equations:

```

DELAY1=LV/delay time
LV=INTEG(input-DELAY1,input*delay time)

DELAY1I=LV/delay time
LV=INTEG(input-DELAY1I,initial value*delay time)

```

See also: DELAY3, SMOOTH, SMOOTH3

Units: DELAY1(units,time) --> units
 DELAY1I(units,time,units) --> units

The input units match the output units. The units of delay time must match those of TIME STEP. For DELAY1I units for the initial value must match those of the input.

DELAY3(input,delay time)
DELAY3I(input,delay time, initial value)

3rd order exponential DELAY
3rd order exponential DELAY with Initial

Returns a 3rd order exponential delay of the input, conserving the input if the delay time changes.
 Equivalent to the equations:

```

DELAY3=LV3/DL
LV3=INTEG(RT2-DELAY3,DL*IN)
RT2=LV2/DL
LV2=INTEG(RT1-RT2,LV3)
RT1=LV1/DL
LV1=INTEG(input-RT1,LV3)
DL=delay time/3

DELAY3I=LV3/DL
LV3=INTEG(RT2-DELAY3I,initial value*DL)
RT2=LV2/DL
LV2=INTEG(RT1-RT2,LV3)
RT1=LV1/DL
LV1=INTEG(input-RT1,LV3)

```

See also: SMOOTH, SMOOTH3

Units: DELAY3(units,time) --> units
 DELAY3I(units,time,units) --> units

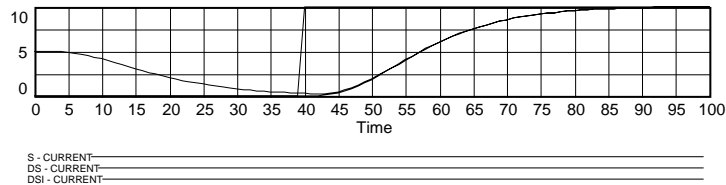
The input units match the output units. The units of delay time must match those of TIME STEP. For DELAY3I units for the initial value must match those of the input.

Example

```

S = STEP(10,40)
DS = DELAY3(S,20)
DSI = DELAY3I(S,20,5)

```



EXP(X)**EXPonential [variable]**

Returns "e to the power of X".

Same as function, `POWER(e,X)`, where $e=2.718...$

See also: `LN`, `LOG`.

Units: `EXP(dimensionless) --> dimensionless` (the argument must be dimensionless)

Examples

EXP(1.0) is equal to 2.718282.

EXP(0.0) is equal to 1.0.

EXP(-10.0) is equal to 4.540E-5.

EXP(10.0) is equal to 22026.46.

EXP(LN(10.0)) is equal to 10.0 (by definition).

GAME(X)[†]**use X except when running a GAME**

Returns X during normal simulation with X being any expression. When in gaming mode (interactive simulation) the value of the right hand side is left constant until changed by the user. Gaming mode is discussed in detail in Chapter 8.

Restrictions: `GAME` must directly follow the equal sign and not be followed by anything. In the Equation Editor select Variable type Auxiliary and subtype Game.

Units: `GAME (units) --> units`

Units for the input and output must match.

Examples

hiring = GAME((desired workforce - workforce)/adjustment time)

GET 123 CONSTANTS('file','tab','cell')†**GET CONSTANTS from 123**

Returns a number, vector or 2-dimensional array of Constant values by querying Lotus 123 for the values. The number of values brought in is determined by the subscripts that are used in the left hand side.

Restrictions: Must appear directly following the equal sign = and not be followed by anything else.

When a model is checked, GET 123 CONSTANTS queries Lotus 123 for the values. If Lotus 123 is not running Vensim will attempt to start it. If Lotus 123 does not have the specified file open Vensim will try to get Lotus 123 to open it. If Vensim cannot get values from Lotus 123 it reports an error. The values are also queried each time a simulation is made and a warning is given if the values are not obtainable. If Vensim has trouble getting values with this function you should try opening the named file in Lotus 123 first.

All of the arguments to GET 123 CONSTANTS are literals and must be enclosed in single quotes '. **'file'** names a file with complete extension to be used. This can be a 123 file or any other type of file Lotus 123 is capable of opening. If no directory is specified (best practice) Vensim will append directory information for the current model. **'tab'** names the tab that contains the Constants. This function will not work with older versions of 123 that do not support tabs. **'cell'** names the cell that the first Constant value is on. Vensim will read additional cells to the right and down if necessary to get all values.

See also: GET 123 DATA, GET XLS CONSTANTS, GET XLS DATA

Units: GET 123 CONSTANTS is not part of units checking. Specify units for the left hand side variable.

See GET XLS DATA for examples of how to use this function.

NOTE GET 123 CONSTANTS is not available on the Macintosh.

GET 123 DATA('file','tab','time row or col','cell')†**GET DATA from 123**

Returns time series data from Lotus 123 for a Data variable or a vector of Data variables.

Restrictions: Must appear directly following the data equals sign := and not be followed by anything.

The GET 123 DATA function is invoked during simulation setup, before the active simulation begins. If Lotus 123 is not running Vensim will attempt to start it. If Lotus 123 does not have the specified file open Vensim will try to get Lotus 123 to open it. If Vensim cannot get values from Lotus 123 it reports an error. If Vensim has trouble getting values with this function you should try opening the named file in Lotus 123 first.

All of the arguments to GET 123 CONSTANTS are literals and must be enclosed in single quotes '. **'file'** names a file with complete extension to be used. This can be a 123 file or any other type of file Lotus 123 is capable of opening. If no directory is specified (best practice) Vensim will append directory information for the current model. **'tab'** names the tab that contains the Data. This function will not work with older versions of 123 that do not support tabs. **'time row or col'** is either the number of the row containing Time values (Time running across) or the letter of the column containing Time values (Time running down). Note that the spreadsheet file must contain values for Time and these values must be Time values and not those of an alternate Time Base. **'cell'** names the cell that the

19: Vensim PLE User's Guide

first Data value is on. Vensim will read values for different times across or down depending on the '**time row or col**' argument. Vensim will also read additional subscript elements in the other direction.

See also: GET 123 CONSTANTS, GET XLS CONSTANTS, GET XLS DATA.

Units: GET 123 DATA is not part of units checking. Specify units for the left hand side variable.

See GET XLS DATA for examples of how to use this function.

NOTE GET 123 DATA is not available on the Macintosh.

GET XLS CONSTANTS('file','tab','cell')[†]

GET CONSTANTS from XLS

Returns a number, vector or 2-dimensional array of Constant values by querying Microsoft Excel for the values. The number of values brought in is determined by the subscripts that are used in the left hand side.

Restrictions: Must appear directly following the equal sign = and not be followed by anything else.

When a model is checked, GET XLS CONSTANTS queries Microsoft Excel for the values. If Microsoft Excel is not running Vensim will attempt to start it. If Microsoft Excel does not have the specified file open Vensim will try to get Microsoft Excel to open it. If Vensim cannot get values from Microsoft Excel it reports an error. The values are also queried each time a simulation is made and a warning is given if the values are not obtainable. If Vensim has trouble getting values with this function you should try opening the named file in Microsoft Excel first.

All of the arguments to GET XLS CONSTANTS are literals and must be enclosed in single quotes '. '**file**' names a file with complete extension to be used. This can be an Excel file or any other type of file Microsoft Excel is capable of opening. If no directory is specified (best practice) Vensim will append directory information for the current model. '**tab**' names the tab that contains the Constants. This function will not work with older versions of Excel that do not support tabs. '**cell**' names the cell that the first Constant value is on. Vensim will read additional cells to the right and down if necessary to get all values.

See also: GET 123 CONSTANTS GET 123 DATA, GET XLS DATA, TABBED ARRAY

Units: GET XLS CONSTANTS is not part of units checking. Specify units for the left hand side variable.

Example

	A	B	C	D
1	Test for GET...			
2	Cap Cost	100		

Cap Cost = GET XLS CONSTANTS('test.xls','Sheet1','B2') ~~|

GET XLS DATA('file','tab','time row or col','cell')[†]**GET DATA from XLS**

Returns time series data from Microsoft Excel for a Data variable or a vector of Data variables.

Restrictions: Must appear directly following the data equals sign := and not be followed by anything.

The GET XLS DATA function is invoked during simulation setup, before the active simulation begins. If Microsoft Excel is not running Vensim will attempt to start it. If Microsoft Excel does not have the specified file open Vensim will try to get Microsoft Excel to open it. If Vensim cannot get values from Microsoft Excel it reports an error. If Vensim has trouble getting values with this function you should try opening the named file in Microsoft Excel first.

All of the arguments to GET XLS CONSTANTS are literals and must be enclosed in single quotes '. **'file'** names a file with complete extension to be used. This can be an Excel file or any other type of file Microsoft Excel is capable of opening. If no directory is specified (best practice) Vensim will append directory information for the current model. **'tab'** names the tab that contains the Data. This function will not work with older versions of Excel that do not support tabs. **'time row or col'** is either the number of the row containing Time values (Time running across) or the letter of the column containing Time values (Time running down). Note that the spreadsheet file must contain values for Time and these values must be Time values and not those of an alternate Time Base. **'cell'** names the cell that the first Data value is on. Vensim will read values for different times across or down depending on the **'time row or col'** argument. Vensim will also read additional subscript elements in the other direction.

See also: GET 123 CONSTANTS, GET 123 DATA, GET XLS CONSTANTS.

Units: GET XLS DATA is not part of units checking. Specify units for the left hand side variable.

Example

	A	B	C	D	E
1	Time	0	10	80	100
2	Profit	8	9	6	12

```
profit := GET XLS DATA('test.xls','Sheet2','1','B2') ~~|
```

Note that the variable names do not need to match and that the order of subscripts determines how the data in the spreadsheet are interpreted.

IF THEN ELSE(cond, tval, fval)**Traditional If-Then Statement**

Returns first value (tval) if condition (cond) is true; second value (fval) if condition is false. cond must be a Boolean expression or an expression or variable that can be interpreted as Boolean. Only the value returned is evaluated, so the other value could be an expression that would lead to an error.

Units: IF THEN ELSE(dimensionless,⁵ units, units) --> units

Examples

IF THEN ELSE(1.0<2.0, 3.0, 4.0) is equal to 3.0.
IF THEN ELSE(1.0>2.0, 3.0, 4.0) is equal to 4.0.
IF THEN ELSE(X = 0.0, 1.0, 1.0 / X) is equal to 1/X unless X is 0.0 when it is equal to 1.0. If X is 0.0, Vensim will not try to compute 1/X and there will be no error.

INITIAL(A)**INITIAL value[variable]**

Returns the value A at initialization and does not change it during a simulation. INITIAL is used to record and hold or "remember" a variable's starting value.

Restrictions: Must appear first on the right of the = sign. It defines a variable as being an Initial variable.

See also: REINITIAL, SAMPLE IF TRUE

Units: Initial (unit) --> unit

Example

x init = INITIAL (x)

x init is set equal to the first value of x.

⁵Expressions such as (a>b) require that a and b have the same dimension and the resulting expression is considered to be dimensionless.

INTEG (rate, initial value)**Numerical INTEGRation**

Returns the integral of the rate. The rate is numerically integrated. The initial value is the value of the variable on the left-hand side of the equation at the start of the simulation.

Restrictions: **INTEG** must directly follow the equal sign. It signals Vensim that the variable on the left-hand side of the equation is a Level or State variable. In the Equation Editor selecting Variable type Level, subtype Normal (the default for variables with boxes around them) will automatically select the **INTEG** function.

Units: `INTEG (unit / time, unit) --> unit`

The units of the integral must be the same as the units of the initial condition. The rate must have the same units, divided by the units of `TIME STEP`.

Examples**Valid**

```
L = INTEG( R * A, 0.0 )
L = INTEG( MAX( A, B ), C )
```

Invalid

```
L = A + INTEG( R, 0.0 )
L = INTEG( B, 0.0 ) + 1
L = 2.0 * INTEG( R, 0.0 ) + 1.0
```

INTEG must follow the equal sign, and it cannot be part of a more complex mathematical expression. These formulations are made valid by defining an auxiliary variable to perform the indicated operations, e.g.

```
L = INTEG( R, 0.0 )
aux = A + L
```

INTEGER(x)**INTEGER part of x**

Returns the largest integer smaller than or equal to X. Same as `QUANTUM(X,1)`.

See also: `QUANTUM`

Units: `INTEGER(units) -> units`

Output and input units must match.

Example

```
INTEGER(5.4) is 5.0
1.5, -.5
```

LN(X)

Natural Logarithm of [variable]

Returns the natural logarithm of X.

Same as LOG(X, 2.718282).

See also: LOG, EXP, POWER.

Units: LN(dimensionless) --> dimensionless (the argument must be dimensionless)

Examples

LN(2.718282) is equal to 1.0.

LN(-5.0) causes an error (LN of a number <= 0 is not defined).

LN(EXP(2.0)) is equal to 2.0 (by definition).

MAX(A,B)

MAXimum of Two Alternatives

Returns the larger of A and B.

Same as IF THEN ELSE(A > B, A, B).

See also: MIN.

Units: MAX(unit, unit) --> unit (all arguments must have the same units)

Examples

MAX(1,2) is equal to 2.0.

MAX(1,1) is equal to 1.0.

MIN(A,B)

MINimum of Two Alternatives

Returns the smaller of A and B.

Same as IF THEN ELSE(A < B, A, B).

See also: MAX.

Units: MIN(unit, unit) --> unit (all arguments must have the same units)

MODULO(A,B)**MODULO of A relative to B**

Returns the remainder when A is divided by B.

Same as A-QUANTUM(A,B).

See also: INTEGER

Units: MODULO(unit, unit) --> unit (all arguments must have the same units)

Examples

MODULO(9,5) is equal to 4.0.

MODULO(76.5,70) is equal to 6.5.

MODULO(8.3,7.3) is equal to 1.0

PULSE(start,width)**PULSE**

Returns 1.0, starting at time start, and lasting for interval width; 0.0 is returned at all other times.

Same as:

```
IF THEN ELSE (time plus > start
  :AND: time plus < (start + width)),1.0,0.0 )
time plus = Time + ( TIME STEP / 2.0 )
```

With PULSE, Vensim Creates time plus internally to avoid rounding errors in comparing Time with start+width.

NOTE The value returned by PULSE does not change except at TIME STEP intervals regardless of the integration technique used.

Units: PULSE(time, time) --> dimensionless (start and width have the same units as Time, the result of PULSE is dimensionless)

Example

```
task active = PULSE( task start, task duration )
```

RAMP(slope,start time,end time)**RAMP test input**

Returns 0 until the start time and then slopes upward until end time and then holds constant.

Same as:

```
IF THEN ELSE ( Time > start time ,
    IF THEN ELSE ( Time < end time :AND: end time > Time ,
        slope*(Time - start time),
        slope*(end time - start time),
    0 )
```

NOTE The value returned by RAMP does not change except at TIME STEP intervals regardless of the integration method used.

See also: STEP, PULSE

Units: RAMP(units,time, time) --> units*time (start time and end time have the same units as Time, the result of RAMP has the product of the slope and Time)

Example

RAMP(1,10,25) is 0 till time 10, then a line to 15 at time 25, then 15 afterwards.

RANDOM NORMAL(m,x,h,r,s)
RANDOM UNIFORM(m,x,s)
NORMAL with mean 0 and standard deviation 1
UNIFORM between m and x

Each of these routines returns a random number. The number returned is different on each successive invocation. These functions are used to introduce noise into a simulation.

Arguments

m is the minimum value that the function will return. Where necessary the distributions will be truncated to return values above this. Truncation occurs after the output has been stretched and shifted.

x is the maximum value that the function will return. Where necessary the distributions will be truncated to return values below this. Truncation occurs after the output has been stretched and shifted.

s is a seed for the distribution to use. If **s** is set to 0 the default noise stream will be used. The default noise stream can be controlled using the NOISE SEED variable described below. For each distinct non-zero value of **s** a separate noise stream will be created. You can couple noise streams by giving them the same seed, but such streams will not be the same. See the examples below.

NOTE The noise seed for a random variable should be a number or a constant. If you make the seed a variable a new noise stream will be started each time the value of that variable changes.

Units: RANDOM...(units,units...dmnl) -> units

The minimum, maximum arguments should have the same units and the RANDOM functions will return these units. If there is a shift argument it should also have these units. The seed argument should be dimensionless. If there is a stretch argument it should be dimensionless. Except as noted

below, the remaining arguments should be dimensionless. If all inputs are numbers the output will be dimensioned by usage.

Specifics

RANDOM NORMAL(m,x,h,r,s) provides a normal distribution of mean 0 and variance 1 before it is stretched, shifted and truncated. This is equivalent to a normal distribution with mean **h** and standard deviation **r**. The units of **r** should match **m**, **x** and **h**.

RANDOM UNIFORM(m,x) provides a uniform distribution between m and x.

NOISE SEED

When the seed passed to the RANDOM functions is zero, they will use the default noise seed. You can control this by creating a variable in the model (usually a constant) called NOISE SEED. When this variable exists in the model its value is used to initialize the noise streams. Changing NOISE SEED will then generate alternative noise streams in different simulations.

Examples

```
driving error1 = RANDOM NORMAL(0,20,12,5,2)
driving error2 = RANDOM NORMAL(0,20,12,5,2)
```

Will generate two distinct noise streams with the same statistical characteristics. Adding

```
driving error3 = RANDOM UNIFORM(0,20,2)
```

to the first two will change the first two noise streams. In contrast adding

```
driving error3 = RANDOM UNIFORM(0,20,0)
```

to the first two equations will leave the first two noise streams unchanged.

```

RC COMPARE('runname',var,mult[,start[,duration]])
RC COMPARE CHECK('runname',var,grace,mult[,start[,duration]])
RC DECAY(basis,decaytime[,start[,duration]])
RC DECAY CHECK(grace,basis,decaytime[,start[,duration]])
RC GROW(basis,growrate[,start[,duration]])
RC GROW CHECK(grace,basis,growrate[,start[,duration]])
RC RAMP(basis,mult,ramptime[,start[,duration]])
RC RAMP CHECK(grace,basis,mult,ramptime[,start[,duration]])
RC STEP(basis,mult[,start[,duration]])
RC STEP CHECK(grace,basis,mult[,start[,duration]])

```

The RC and RC CHECK functions all work in the same manner. Each keeps a variable at its normally generated model value until a specified time, and then defines a new trajectory. The RC functions are used in test inputs (in :THE CONDITION: part of a Reality Check) and the RC CHECK functions are used in the consequence (:IMPLIES:) portion of a Reality Check. Each of the functions allows you to specify the time at which the change to a new trajectory should occur, or use the value of the model variable RC START TIME to begin the test. Normally RC START TIME is a constant that is set to a time shortly after the beginning of the simulation. If RC START TIME is not present in the model the trajectory changes are made at time INITIAL TIME + TIME STEP. Chapter 18 contains more information on Reality Check.

Restrictions: The RC functions can not be used outside of Reality Check equations. They are valid only for comparison to a variable and must be used in the form:

```

rc example
:THE CONDITION: var1 = RC STEP(var1,0)
:IMPLIES: var2 <= RC STEP CHECK(short grace,var2,0)

```

Note that each of the RC functions has two optional arguments **start** and **duration**. **start** is the time at which the trajectory change takes place, and **duration** the length of time the trajectory remains changed. If **duration** is missing the trajectory change will continue to the end of the simulation. If **start** is missing the trajectory change will start at RC START TIME as described above.

The following arguments are common to all, or almost all, the functions:

basis is the value against which the trajectory change will take place and will be multiplied by **mult** when the trajectory change begins. **basis** can be an expression, though normally it will just be the name of the variable on the left hand side of the comparison operator as it is in the above example. The value that **basis** takes on at RC START TIME is used to perform the trajectory changes (except for the RC COMPARE functions).

grace is the amount of time that can pass before the RC...STEP function actually begins checking for compliance. This makes it easy to specify that an effect takes place after a delay. Note that the value of **basis** is frozen at RC START TIME so that RC STEP CHECK(5,inventory,.3,5) and RC STEP CHECK(0,inventory,.3,10) have quite different meanings.

mult determines the amount that **basis** will be multiplied by to get the new value. This can be an expression, though in many cases it will just be a number such as 0 or 2. Unlike **basis**, the current value of **mult** is used at all times and thus can provide arbitrary time profiles.

```

RC COMPARE( 'runname', var, mult [,start[,duration]])
RC COMPARE CHECK( 'runname', var, grace, mult [,start[,duration]])

```

compares the values of a variable from a different run - normally a base case. **var** replaces **basis** and can only be a variable name, not an expression. '**runname**' is a single quoted literal or a string variable that names the run against which the comparison will be made. The COMPARE functions useful for Constraints that say things like if we raise our price sales will be smaller than they would have been.

RC DECAY(basis, decaytime [,start[,duration]])

RC DECAY CHECK(grace, basis, decaytime [,start[,duration]])

forces an exponential decay to zero with a time constant **decaytime**. This function is useful when you want to make sure something goes to zero, but you don't care if it does so as an exponential decay and therefore would technically never actually get to zero.

RC GROW(basis, growrate [,start[,duration]])

RC GROW CHECK(grace, basis, growrate [,start[,duration]])

forces exponential growth at **growrate**. This is useful for making sure variables grow sufficiently fast.

RC RAMP(basis , mult , ramptime [,start[,duration]])

RC RAMP CHECK(grace, basis, mult, ramptime [,start[,duration]])

force a ramp to **basis*mult** over **ramptime**. This is a more continuous change than RC STEP and is useful for checking to see that variables make monotone adjustments to a new value. The value of the RC RAMP functions after RC START TIME is **basis*(mult*(Time-RC START TIME)/ramptime + (1-(Time-RC START TIME)/ramptime))** until **RC START TIME + ramptime** when it just takes on the value **basis*mult**.

RC STEP(basis, mult [,start[,duration]])

RC STEP CHECK(grace, basis, mult [,start[,duration]])

forces a step to a new trajectory at RC START TIME. A step is a strongly disruptive shock to a system and has, therefore, been the most commonly used. Step changes are abrupt and will often cause formulations that are not completely robust to fail. The idiomatic way to use the RC STEP function is:

```
var = RC STEP(var, .5)
```

This causes the variable to jump to 50% of its value at RC STEP TIME and then remain constant. Note that **mult** may be time varying, but that **basis** is evaluated only at the step time (more precisely one TIME STEP prior to the step time). The idiomatic way to use the RC STEP CHECK function is:

```
no pop no production :THE CONDITION:
  population = RC STEP(population,0) :IMPLIES:
    production <= RC STEP CHECK(1,production,0)
```

This would drop *population* to 0 at *RC STEP TIME* and then check that *production* goes to zero 1 year (assuming the model is in years) thereafter.

Units: The RC functions are not part of units checking.

SIN(X)**SINE of X**

Returns the sine of X. Sometimes useful to test the dynamic response of a system. SIN is periodic on X in the range of 0 to 2 pi radians.

Units: SIN (dimensionless) -> dimensionless

Examples

sin(0.0) is equal to 0.0.

sin(1.0) is equal to 0.84147.

SMOOTH(input,delay time)**exponential SMOOTH****SMOOTHI(input,delay time,initial value)****exponential SMOOTH with Initial**

Returns a exponential smooth of the input. Equivalent to the equations:

SMOOTH=INTEG((input-SMOOTH)/delay time,input)

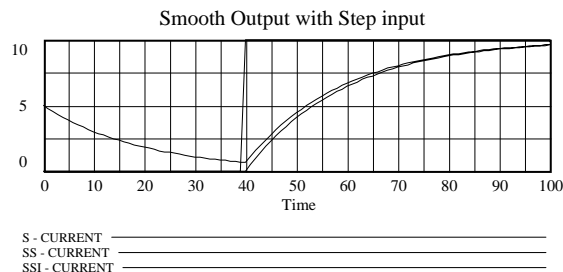
SMOOTHI=INTEG((input-SMOOTHI)/delay time,initial value)

See also: DELAY1, DELAY3, SMOOTH3, DELAYP

Units: SMOOTH(units,time) --> units

SMOOTHI(units,time,units) --> units

The input units match the output units. The units of delay time must match those of TIME STEP. For SMOOTHI units for the initial value must match those of the input.

Example

S = STEP(10,40)

SS = SMOOTH(S,20)

SSI = SMOOTHI(S,20,5)

SMOOTH3(input,delay time)
SMOOTH3I(input,delay time,initial value)

3rd order exponential SMOOTH
3rd order exponential SMOOTH with Initial

Returns a 3rd order exponential smooth of the input. Equivalent to the equations:

$SMOOTH3 = INTEG((LV2 - SMOOTH3)/DL, input)$	$SMOOTH3I = INTEG((LV2 - SMOOTH3I)/DL, initial\ value)$
$LV2 = INTEG((LV1 - LV2)/DL, input)$	$LV2 = INTEG((LV1 - LV2)/DL, initial\ value)$
$LV1 = INTEG((IN - LV1)/DL, input)$	$LV1 = INTEG((IN - LV1)/DL, initial\ value)$
$DL = delay\ time/3$	

See also: DELAY3, SMOOTH, DELAYP

NOTE The SMOOTH3 function does not conserve material when the delay time is changing. It is intended to be used for information delays.

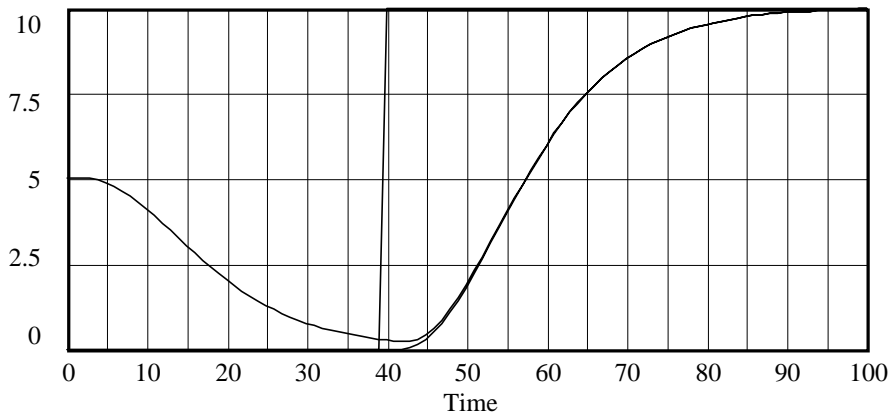
Units: SMOOTH3(units,time) --> units
 SMOOTH3I(units,time,units) --> units

The input units match the output units. The units of delay time must match those of TIME STEP. For SMOOTH3I units for the initial value must match those of the input.

Example

```
S = STEP(10,40)
SS = SMOOTH3(S,20)
SSI = SMOOTH3I(S,20,5)
```

Smooth3 Output with Step input



S - CURRENT _____
 SS - CURRENT _____
 SSI - CURRENT _____

SQRT(X)**SQuare RooT of X**

Returns the square root of X.

Same as: `POWER(X,0.5)`.

Units: `SQRT(units*units) --> units`

If argument has units that are a perfect square the result will be the square root of the units. If the argument is dimensionless, the result should be dimensionless.

Units:Example: `SQRT(9.0)` is equal to 3.0.

STEP(height,step time)**STEP test input**

Returns 0 until the step time and then returns height.

Same as:

```
IF THEN ELSE ( Time plus > step time,height,0)
time plus = Time + ( TIME STEP / 2.0 )
```

NOTE The value returned by STEP does not change except at TIME STEP intervals regardless of the integration method used.

See also: RAMP, PULSE

Units: `STEP(units,time) --> units` (step time has the same units as Time, the result of STEP has the units of the step height)

Example

`STEP(10,20)` is 0 till time 20, then 10.

TIME TRANSITION(x1,x2,...)**Perform a TIME TRANSITION**

The TIME TRANSITION function is obsolete. Use the RC ... functions instead.

WITH LOOKUP(x,(L#))**LOOKUP the y value in the xy pairs L# corresponding to x.**

Specifies a nonlinear relationship between the input **x** and the output by passing the input through a series of x,y pairs specified as numbers. It is the same as **L(x)** where **L** is defined by the equation **L(L#)**.

Restrictions: Must appear first on the right hand side of the equation and can not be followed by anything else.

The WITH LOOKUP function is a convenient way to specify a nonlinear relationship without explicitly naming the Lookup function to be used. This can reduce clutter and be somewhat quicker than naming a separate Lookup variable but is not as flexible.

Units: WITH LOOKUP(dmn1,#) --> units

Just as for normal Lookup usage the x argument is expected to be dimensionless. If not, a warning is issued, but not an error. The output units are those for the left hand side variable.

Example

X=WITH LOOKUP(1.5,((0,1),(1,1),(2,2)) is equal to 1.5.

XIDZ(A,B,X)**X If Divided by Zero (otherwise A/B)**

Returns **A** divided by **B**. If **B** is zero, then returns **X**. XIDZ is normally used to express some limit of **A/B**, as **B** approaches 0 (which would normally be undefined for **B = 0**).

Same as: IF THEN ELSE (**B = 0** , **X** , **A/B**)⁶.

Units: XIDZ(units of A, units of B, units of A/B) --> units of A/B (i.e., same unit behavior as the division operator)

Examples

XIDZ(3, 4, 1) is equal to 0.75.

XIDZ(3, 0, 1) is equal to 1.0.

ZIDZ(A,B)**Zero If Divided by Zero (otherwise A/B)**

Divide **A** by **B**. If **B** is zero, then return 0.0. ZIDZ is normally used to express the special case where the limit of **A/B**, as **B** approaches 0, is 0.

Same as: XIDZ(**A** , **B** , 0.0) .

Units: ZIDZ(units of A, units of B) --> units of A/B (i.e., same unit behavior as the division operator)

Examples

ZIDZ(3, 4) is equal to 0.75.

ZIDZ(3, 0) is equal to 0.

⁶Both XIDZ and ZIDZ actually test the absolute value of B against a small (1E-6) number and use X if B is less than this small number.

Lookups

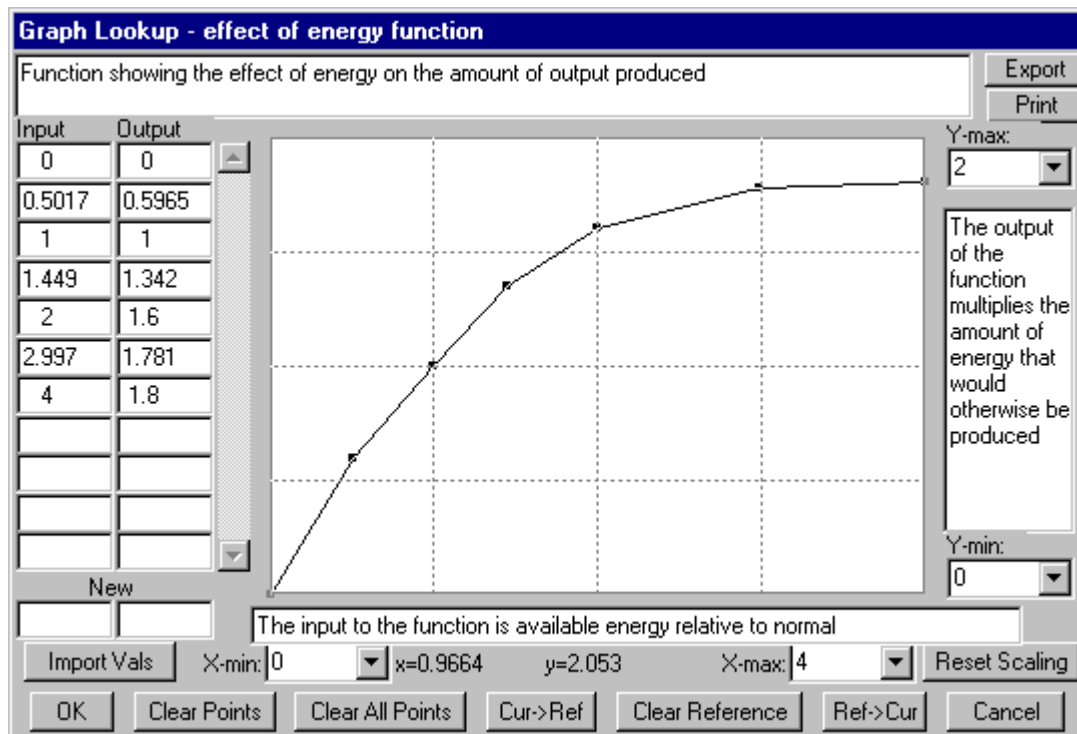
In addition to the predefined functions, you can specify an arbitrary nonlinear relationship with a Lookup. Put simply, a Lookup is a list of numbers representing an x axis and a y axis. The inputs to the Lookup are positioned relative to the x axis, and the output is read from the y axis. You use Lookups to create your own specialized functions.

Lookups are also referred to as "Lookup Functions," "Tables," "Lookup Tables," "Table Functions," and sometimes "Graphical Functions."

Lookups can be declared as x,y pairs or by specifying the x-axis followed by the y-axis. The format for declaring a Lookup is:

```
LOOKUP NAME( [ (Xmin,Xmax)-(Ymin,Ymax), (Xref1, Yref1), (Xref2,Yref2), . . . (Xrefn,Yrefn) ]
(X1, Y1), (X2,Y2), . . . (Xn, Yn) )
~ Units ~ Description |
```

Lookups are most commonly defined using the Graph Lookup Editor:



The Graph Lookup Editor allows you to enter x,y pairs, or to trace the shape of the Lookup using the mouse. The Graph Lookup Editor is described in detail in Chapter 6.

When you use the Graph Lookup Editor, it will put additional information on scaling in the lookup definition as in:

```
effect of energy function([(0,0)-(4,2)],(0,0),(0.501742,0.596491),
(1,1),(1.44948,1.34211),(2,1.6),(2.99652,1.7807),(4,1.8))
```

The range shown in square brackets [(0,0)-(4,2)] is used by the Graph Lookup Editor to determine the scaling. It can also be followed by optional reference point information. There is no need to enter a scale or reference point information when typing in Lookup values.

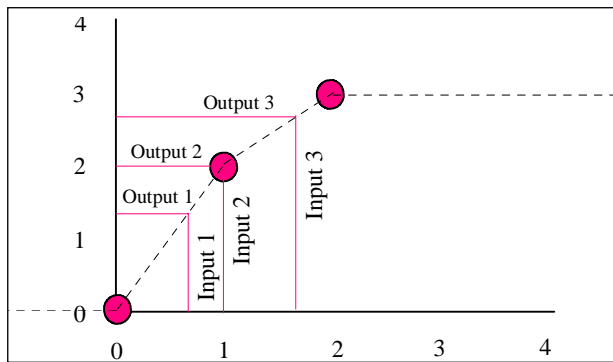
Using Lookups

The use of a Lookup function is the same as that of the predefined functions that take one variable, as in:

```
outvar = lookup name(invar) ~~|
```

Units NOTE: Variables coming into a Lookup function are recommended to be dimensionless, and a warning message is given during units checking if they are not. The output of a Lookup function has the dimensions specified in the definition of the Lookup.

When used in the above manner the value of the output is determined by linear interpolation of the lookup, as in:



Here the dark circles represent the points on the Lookup, corresponding to $((0, 0), (1, 2), (2, 3))$. The dashed line connecting the points shows how the lookup is interpolated. The dashed lines to the right and left show that if the input is below 0, the output will be 0, and if the input is above 2, the output will be 3. Three sample inputs (0.67, 1.3, 2.0) and their respective outputs (1.3, 2.0, 2.7) are shown.

Lookups in Vensim will always hold the first or last value when the input goes outside the range of the Lookup. You will receive a warning message whenever this happens.

If you wish to have Lookups extrapolate, you should use the LOOKUP EXTRAPOLATE function described earlier in this chapter, or define a large minimum or maximum input value as in:

```
extrap up lookup ((0,0),(0.5,7),(0.75,.9),(1,1),
(2,2),(1E6,1E6))
```

19: Vensim PLE User's Guide

Example 1

```
effect energy tab (
  (0.0,0.0), (0.1,0.2), (0.2,0.4),(0.3,0.6), (0.4,0.7), (0.5,0.8),
  (0.6,0.85), (0.7,0.9), (0.8,0.95), (0.9,0.98), (1.0,1.0),
  (1.2,1.02), (2,1.02) )
~dimensionless
~ The effect of energy availability on industrial production.
/
production = nominal production
  * effect energy tab( US energy
  * fraction available - energy losses )
  ~ quads / year
  ~ energy-availability limited production rate
/
```

Example 2

```
effect energy tab(0, 0.5, 1.0, 2.0,
  0, 0.7, 1.0, 1.1)
  ~ dimensionless
  ~ The effect of energy availability on industrial production.
/
production = nominal production
  * effect energy tab( energy availability )
  ~ quads / year
  ~ production rate limited by energy availability
/
```

The x,y pairs for a Lookup can be altered at runtime, and the number of x,y pairs specified in a Lookup can be increased or decreased.

Appendix A—Models that Come with Vensim PLE

Appendix A outlines the name and location of the models developed in the chapters of this manual, and also describes the sample models included with Vensim that are not developed in the manual.

Some of the sample models contain multiple views. When you open a model with multiple views in Vensim PLE, a message informs you of the fact. Vensim PLE Plus will open models with multiple views without comment. The name of the View you are working with appears in the Status Bar at the bottom of the window. Click on this name to select another view to work with. You can also use Page Up and Page Down to move through views.

Chapter Models

The Vensim PLE Chapter models will install into *chap*** subdirectories of the *plemodel* subdirectory. The models you can create by working through this *User's Guide* are contained in *complete* subdirectories, so that you can create your own models in the *chap*** subdirectories, and refer to the solutions if you choose. Models that you can use (but which lack detailed instructions on construction) are contained directly in the *chap*** subdirectories. The normal complete path for models is *c:\Program Files\Vensim\plemodel\...* but you can choose another directory to install Vensim PLE into.

Sample Models

The Sample models install into the *sample* subdirectory of the *plemodel* subdirectory of Vensim. The full path for this directory is normally *c:\vensim\plemodel\sample*.

burnout.mdl

This is a worker burnout model written by Jack Homer and published in *The System Dynamics Review* Volume 1, Number 1, 1985, Pages 42-62.

commod.mdl

A commodities model of hog breeding only slightly different from that developed by Dennis Meadows in *Dynamics of Commodity Production Cycles* (The MIT Press, Cambridge MA 1970 - available from Pegasus Communications).

corpgrth.mdl

The corporate growth model is a simple example of resource-limited growth of a company. This model was developed and published by Jay W. Forrester in the book *Principles of Systems* (The MIT Press, Cambridge, MA, 1968 - available from Pegasus Communications).

epidemic.mdl

This model shows the spread of an epidemic by means of a Bass diffusion model. The diffusion structure can be used in many different areas and an example is shown in Chapter 5 (*customer.mdl*).

procrast.mdl

This is a simple model of the procrastination process. It is closely related to project models.

projexmp.mdl

This is a very simple project model example.

proj.mdl

This is a simple project model and is very nearly the same as that developed in *Introduction to System Dynamics Modeling with DYNAMO*, by G.P. Richardson and A.L. Pugh (The MIT Press, Cambridge, MA, 1981 - available from Pegasus Communications).

urban.mdl

This is the model of urban development and decay presented in *Urban Dynamics* by J.W. Forrester (The MIT Press, Cambridge, MA, 1969 - available from Pegasus Communications).

world.mdl

This is the model presented in *World Dynamics* by Jay W. Forrester (The MIT Press, Cambridge, MA, 1971; second edition, 1973 - available from Pegasus Communications).

wrld3-91.mdl

This is an updated version of the model that was used for the book *The Limits to Growth* by D.H. Meadows et al (Universe Books, New York, 1972). The model was updated for *Beyond the Limits: confronting global collapse, envisioning a sustainable future* by D.H. Meadows et al (Chelsea Green Publishing Company, Post Mills, VT, 1992).

Appendix B—Information Resources

This appendix contains a list of suggested resources in the field of System Dynamics and computer simulation. This list is by no means complete, but provides a place to start and gives pointers to other resources. Almost all books (and many audio and video tapes) published in the field of System Dynamics can be purchased from Productivity Press. Their address and phone number appear later in this appendix.

Books

System Dynamics

System Dynamics is a field that resulted from the pioneering efforts of Jay W. Forrester to apply the engineering principles of feedback and control to social systems. One of the earliest, and still one of the best references in this field is *Industrial Dynamics* by Jay W. Forrester (The MIT Press,⁷ Cambridge, MA, 1961). A simpler introduction to System Dynamics, along with a number of exercises, by the same author, is *Principles of System* (The MIT Press, Cambridge, MA, 1968) available from Pegasus Communications.

A more recent textbook which gives a straightforward and very readable presentation of how to build models is *Introduction to System Dynamics Modeling with DYNAMO*, by G.P. Richardson and A.L. Pugh (The MIT Press, Cambridge, MA, 1981) available from Pegasus Communications.

Another useful resource, containing a number of interesting real world examples is *System Enquiry, A System Dynamic Approach* by Eric F. Wolstenholme (John Wiley & Sons, New York 1990).

A very popular and readable book covering many conceptual issues in thinking systemically about problems is *The Fifth Discipline* by Peter M. Senge (Doubleday, New York, 1990).

An interesting and controversial work *World Dynamics* by Jay W. Forrester (The MIT Press, Cambridge, MA, 1971; second edition, 1973, available from Pegasus Communications) discusses growth in a finite world. The World model contained in this book is included in the models supplied with Vensim.

Following up on *World Dynamics* was the widely read book *The Limits to Growth* by Donella H. Meadows, Dennis L. Meadows, Jørgen Randers and William W. Behrens, III (Universe Books, New York 1972) and a follow on work *Beyond the Limits* by Donella H. Meadows, Dennis L. Meadows and Jørgen Randers (Chelsea Green Publishing Company, Post Mills Vermont 1992). Detailed discussion of the model used in these works is presented in *Dynamics of Growth in a Finite World* by Dennis L. Meadows et al (The MIT Press, Cambridge MA 1974). The world3-91 model in *Beyond the Limits* is included with Vensim.

⁷All of the MIT Press books mentioned in this bibliography are available from Productivity Press.

Appendix B: Vensim PLE User's Guide

Modeling for Learning Organizations. Eds John D. W. Morecroft and John D. Sterman, Productivity Press, Portland, OR, 1994. Covers some of the recent work in modeling businesses and other organizations, and includes case studies and reviews of simulation software.

Urban Dynamics by Jay W. Forrester (Productivity Press, Portland, OR, reprinted) covers urban development, with emphasis on job training programs, housing and employment issues. The urban model in *Urban Dynamics* is included with Vensim.

Other Areas

Living in the Environment, 9th Edition by G. Tyler Miller Jr. (Wadsworth Publishing Company, Belmont CA 1996 ISBN 0-534-23898-X) is a biology and ecology textbook that includes discussion of a number of issues from a system dynamics perspective. A companion to this work, *Simulations for Miller's Living in the Environment 9th Edition*, by David Peterson (Wadsworth Publishing Company, Belmont CA 1996 ISBN 0-534-23905-6) presents a number of models demonstrating issues discussed in the book. Both are available from bookstores or from Thomson Communications, 7625 Empire Drive, Florence, KY 41042 Phone: 800-865-5840, Fax: 606-647-5013.

Roadmaps

Roadmaps is a series of writings and exercises that are intended as a self study guide to system dynamics. They have been developed by MIT undergraduates working under the direction of Jay W. Forrester. The material is available for free from the MIT system dynamics in education web site (<http://sysdyn.mit.edu>) and can also be purchased for a nominal charge from the Creative Learning Exchange (contact information below). Some of the mechanical activities in Roadmaps are not applicable to Vensim PLE but most of the conceptual content is relevant.

There is also a distance learning course based on the Roadmaps that is being offered on an experimental basis (contact Nan Lux nlux@mit.edu for more information on this).

Publishers

Pegasus Communications publishes a range of system dynamics books in addition to those already mentioned. They also publish *The Systems Thinker* and other material with a focus on learning organizations. They have a catalogue of books and other materials.

Pegasus Communications, Inc.
One Moody Street
Waltham MA 02154-5339
Phone: 781 398 9700
Fax: 781 894 7175
<http://www.pegasuscomm.com>

The Creative Learning Exchange is a non profit organization devoted to the dissemination of materials to improve learning in K-12 education, with a special focus on system dynamics:

Creative Learning Exchange
1 Keefe Road
Acton, Massachusetts 01720
Phone: 978 287 0070
Fax: 978 287 0080

System Dynamics Society

The System Dynamics Society is an international, nonprofit organization devoted to encouraging the development and use of system dynamics around the world. It is run solely on a volunteer basis by practitioners of System Dynamics in academics and industry. With members in over 35 countries the System Dynamics Society provides a forum in which researchers can keep up to date with applications, methodologies and tools.

The System Dynamics Society is responsible for the publication of *The System Dynamics Review*, a refereed journal containing articles on methodology and application. The International Conference of the System Dynamics Society is held annually. This meeting brings together people from around the world and is arranged to give broad exposure to the local activities of the area hosting it. A President's letter provides an informal forum to bring members up to date on society activities and business.

In addition to *The System Dynamics Review* and its annual conferences, the Society maintains supplies of past conference proceedings and other important publications in the field. An electronic bibliography of work done in the field is maintained and updated periodically. The "Beer Game," a production distribution board game illustrating important principles relating local decision making to system behavior is available along with video tape and instructional material. To order the Beer Game or other publication contact:

Roberta Spencer, Executive Director
The System Dynamics Society
Milne 300, University at Albany
135 Western Avenue
Albany NY 12222

Phone: (518) 442 3865
Fax: (518) 442-3398
Email: system.dynamics@albany.edu

The System Dynamics Society is open to all individuals. The annual membership fee is \$90.00, or \$45.00 for full time students (please attach proof of enrollment) and individuals with a gross family annual income of less than US\$20,000.00 (please attach a short note confirming your status). Membership includes a subscription to the System Dynamics Review. Send a check with your name and contact information to:

John Wiley & Sons
Periodicals Division
P.O. Box 7247-8491
Philadelphia PA 19170-8491

or

John Wiley & Sons
Baffins Lane
Chichester
Sussex PO19 1UD
ENGLAND

Internet Resources

World Wide Web

The World Wide Web contains many sites of interest to system dynamics and other simulation modeling. The sites listed here all contain links to other sites.

The Vensim Home Page (<http://www.vensim.com>) contains information about Vensim and has a number of downloadable programs including Vensim PLE.

The system-dynamics mailing list (see below) home page (<http://www.vensim.com/sdmail.html>) contains information about system dynamics and also provides access to the system dynamics bibliography.

The MIT System Dynamics in Education Project home page (<http://sysdyn.mit.edu>) provides a list of resources that are useful for people in education and also contains a wealth of material that has been done internally at MIT but never published.

Mailing Lists

system-dynamics

The system-dynamics mailing list is a moderated internet mailing list for discussions of issues in system dynamics. When someone posts a message to the list, it is rebroadcast to all list subscribers (a digest form is also available) To subscribe, send email to majordomo@world.std.com with the message in the main body of text

```
subscribe system-dynamics  
end
```

If you would like the digest form, which puts a handful of messages into a digest and then posts them, send the following message

```
subscribe system-dynamics-digest  
end
```

K-12

The K-12 mailing list is a list devoted to discussion of issues related to the use of system dynamics in K through 12 education. If you are interested contact Nan Lux (nlux@mit.edu) and ask, in English, to be added to this list.

License and Support

Vensim PLE Software License

By Clicking on the "Accept" button you are consenting be bound by this agreement. If you do not agree to all of the terms of this agreement click on the "Cancel" button and do not install the software. If you do not agree to the terms of the terms herein you may return the software to the place of purchase a refund of license fees paid.

VENSIM® PLE SOFTWARE LICENSE

This License Agreement has three parts. Parts I and III apply when you have not purchased a license to Vensim PLE (the "SOFTWARE"). Parts II and III apply when you have purchased a license to Vensim PLE (the "SOFTWARE").

PART I—TERMS APPLICABLE WHEN LICENSE FEES NOT (YET) PAID

(Limited to Evaluation and Educational Use).

1. **NONCOMMERCIAL GRANT OF LICENSE.** VENTANA grants you a non-exclusive license to use the Software free of charge if you are using the SOFTWARE for the purposes of learning or teaching system dynamics or systems thinking. You may not use the SOFTWARE for commercial purposes. You may not use, or present the results of using, the SOFTWARE for compensation by an entity that is not a full-time, non-profit, tax-exempt, school, college or university whose primary purpose is to provide instruction to an enrolled body of students through a full-time faculty, licensed by an appropriate state authority, to confer degrees or diplomas which are recognized as qualifying the student to pursue a course of higher education (an "Eligible Institution"). You may not use the software or make use of the results of the software as part of your work unless you are employed by an Eligible Institution.

2. **EVALUATION GRANT OF LICENSE.** VENTANA grants you a non-exclusive license to use the software for the purpose of evaluating whether to purchase an ongoing license to the software. The evaluation period for use by or on behalf of a commercial entity is limited to 90 days. Government Agencies, (other than public libraries) are not considered educational or charitable non-profit organizations for the purposes of this Agreement. If you fit within the description above you, you may use the SOFTWARE in the manner described in Part III below.

3. **DISCLAIMER OF WARRANTY.** Free of charge SOFTWARE is provided on an "AS IS" basis, without warranty of any kind including without limitation the warranties of merchantability, fitness for a particular purpose and non-infringement. The entire risk as to the quality and performance of the Software is borne by you. Should the Software prove defective, you and not VENTANA assume the entire cost of any service and repair. THE FOREGOING WARRANTY AND LIMITATIONS ARE EXCLUSIVE REMEDIES AND ARE IN LIEU OF AND VENTANA EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, CONCERNING THE SOFTWARE OR ANY SERVICES PROVIDED IN CONNECTION THEREWITH, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE. IN NO EVENT WILL VENTANA BE LIABLE FOR SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES, DAMAGES FROM LOSS OF USE, DATA OR PROFITS, OR COST OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, EVEN IF VENTANA SHALL HAVE BEEN ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS, COST OR DAMAGES, ARISING OUT OF OR IN CONNECTION WITH THIS LICENSE OR THE USE OR PERFORMANCE OF THE SOFTWARE, WHETHER IN AN ACTION BASED ON CONTRACT OR TORT INCLUDING NEGLIGENCE.

PART II—TERMS APPLICABLE WHEN LICENSE FEES PAID

4. **LIMITED WARRANTY, DISCLAIMER AND LIMITATION OF LIABILITY.** VENTANA warrants that the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt. VENTANA's entire liability and your exclusive remedy shall be, at VENTANA's option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE that does not meet VENTANA's Limited Warranty and that is returned to VENTANA with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE is (a) due to hardware failure or other cause within your reasonable control; (b) caused by use of the Software other than in accordance with the accompanying documentation or

other instructions provided by VENTANA to you; (c) caused by a failure of, defect in, or change to any software provided by you or a third party operating on or in connection with the SOFTWARE; (d) modified, altered or changed (except by VENTANA); or (e) damaged by accident or misuse or used other than as permitted in this License. THE FOREGOING WARRANTY AND LIMITATIONS ARE EXCLUSIVE REMEDIES AND ARE IN LIEU OF AND VENTANA EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, CONCERNING THE SOFTWARE OR ANY SERVICES PROVIDED IN CONNECTION THEREWITH, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE. IN NO EVENT WILL VENTANA BE LIABLE FOR SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES, DAMAGES FROM LOSS OF USE, DATA OR PROFITS, OR COST OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, EVEN IF VENTANA SHALL HAVE BEEN ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS, COST OR DAMAGES, ARISING OUT OF OR IN CONNECTION WITH THIS LICENSE OR THE USE OR PERFORMANCE OF THE SOFTWARE, WHETHER IN AN ACTION BASED ON CONTRACT OR TORT INCLUDING NEGLIGENCE. IN NO EVENT WILL VENTANA'S TOTAL LIABILITY FOR ANY DAMAGES IN ANY ACTION IN ANY FORM EXCEED THE LICENSE FEES PAID FOR THE LICENSED PROGRAM THAT CAUSED THE DAMAGES.

PART III—TERMS APPLICABLE TO ALL LICENSE GRANTS

5. SCOPE OF GRANT OF LICENSE. VENTANA grants a non-exclusive right to use one copy of the VENTANA software program identified above (the "SOFTWARE"). The right to use the SOFTWARE terminates upon the violation of any provision of this License. Only one copy of the SOFTWARE may be used on a single computer at a time. The SOFTWARE may be transferred for use on another computer, but the SOFTWARE may not be used simultaneously on more than one computer unless additional licenses are purchased for each multiple simultaneous use. The SOFTWARE may be installed on a network, provided that each person accessing the SOFTWARE through the network must have a copy licensed to that person.

6. PROPRIETARY RIGHTS. The SOFTWARE is proprietary to VENTANA and all copyright, patent, trade secret, trademark and other intellectual property rights are and shall remain the valuable property of VENTANA. You may not reverse engineer, decompile, or disassemble the SOFTWARE. You agree to take all necessary steps to ensure that the provisions of this License are not violated by you or by any person under your control or in your service.

7. REPRODUCTION AND COPYRIGHT. The SOFTWARE is protected by United States copyright laws and international treaty provisions. You may copy and freely distribute the Vensim PLE Installation program so long as you do not remove or omit any license, copyright or proprietary information or notices that are part of that program. You may not copy, allow anyone else to copy, or otherwise reproduce any part of the SOFTWARE in any other form without the prior written consent of VENTANA, except that you may (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) copy the SOFTWARE to your office computer and your home or portable computer provided that you comply with the restriction on simultaneous use. You may copy the written material accompanying the SOFTWARE for personal use or for educational use as defined in Paragraph 1. You may make a single copy of the Vensim PLE documentation for commercial use with a licensed copy of the Vensim PLE software.

8. RESTRICTIONS ON REDISTRIBUTION. You may not publicly offer for sale the SOFTWARE or any of the written material accompanying the SOFTWARE without the prior written consent of VENTANA. You may not include the SOFTWARE or the written material accompanying the SOFTWARE in any published work without the prior written consent of Ventana Systems, Inc.

9. GENERAL. This license will be governed by the laws of the Commonwealth of Massachusetts.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS LICENSE AND UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS; YOU FURTHER AGREE THAT IT IS THE COMPLETE STATEMENT OF THE AGREEMENT BETWEEN US WHICH SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US, RELATING TO THE SUBJECT MATTER OF THIS LICENSE.

Vensim PLE Plus Software License

By Clicking on the "Accept" button you are consenting be bound by this agreement. If you do not agree to all of the terms of this agreement click on the "Cancel" button and do not install the software. If you do not agree to the terms of the terms herein you may return the software to the place of purchase for a refund of license fees paid.

VENSIM® PLE PLUS SOFTWARE LICENSE

This License Agreement has three parts. Parts I and II apply when you have purchased an educational license to Vensim PLE Plus (the "SOFTWARE"). Part II applies when you have purchased a commercial license to Vensim PLE Plus (the "SOFTWARE").

PART I—TERMS APPLICABLE WHEN REDUCED LICENSE FEES PAID

(Educational Use).

1. EDUCATIONAL GRANT OF LICENSE. VENTANA grants you a non-exclusive license to use the Software at reduced license fee if you are (i) a full-time, non-profit, tax-exempt, school, college or university whose primary purpose is to provide instruction to an enrolled body of students through a full-time faculty, licensed by an appropriate state authority, to confer degrees or diplomas which are recognized as qualifying the student to pursue a course of higher education (an "Eligible Institution") , or (ii) a full or part-time student at an Eligible Institution, or (iii) a full or part-time faculty member at an Eligible Institution. If you acquire the SOFTWARE under an educational allowance, you may not use the SOFTWARE for commercial purposes or use the SOFTWARE in the provision of consulting services for compensation. If you fit within the description above you, you may use the SOFTWARE in the manner described in Part II below.

PART II—TERMS APPLICABLE TO ALL LICENSE GRANTS

2. LIMITED WARRANTY, DISCLAIMER AND LIMITATION OF LIABILITY. VENTANA warrants that the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt. VENTANA's entire liability and your exclusive remedy shall be, at VENTANA's option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE that does not meet VENTANA's Limited Warranty and that is returned to VENTANA with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE is (a) due to hardware failure or other cause within your reasonable control; (b) caused by use of the Software other than in accordance with the accompanying documentation or other instructions provided by VENTANA to you; (c) caused by a failure of, defect in, or change to any software provided by you or a third party operating on or in connection with the SOFTWARE; (d) modified, altered or changed (except by VENTANA); or (e) damaged by accident or misuse or used other than as permitted in this License. THE FOREGOING WARRANTY AND LIMITATIONS ARE EXCLUSIVE REMEDIES AND ARE IN LIEU OF AND VENTANA EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, CONCERNING THE SOFTWARE OR ANY SERVICES PROVIDED IN CONNECTION THEREWITH, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE. IN NO EVENT WILL VENTANA BE LIABLE FOR SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES, DAMAGES FROM LOSS OF USE, DATA OR PROFITS, OR COST OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, EVEN IF VENTANA SHALL HAVE BEEN ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS, COST OR DAMAGES, ARISING OUT OF OR IN CONNECTION WITH THIS LICENSE OR THE USE OR PERFORMANCE OF THE SOFTWARE, WHETHER IN AN ACTION BASED ON CONTRACT OR TORT INCLUDING NEGLIGENCE. IN NO EVENT WILL VENTANA'S TOTAL LIABILITY FOR ANY DAMAGES IN ANY ACTION IN ANY FORM EXCEED THE LICENSE FEES PAID FOR THE LICENSED PROGRAM THAT CAUSED THE DAMAGES.

3. SCOPE OF GRANT OF LICENSE. VENTANA grants a non-exclusive right to use one copy of the VENTANA software program identified above (the "SOFTWARE"). The right to use the SOFTWARE terminates upon the violation of any provision of this License. Only one copy of the SOFTWARE may be used on a single computer at a time. The SOFTWARE may be transferred for use on another computer, but the SOFTWARE may not be used simultaneously on more than one computer unless additional licenses are purchased for each multiple simultaneous use. The SOFTWARE may be installed on a network, provided that each person accessing the SOFTWARE through the network must have a copy licensed to that person.

4. PROPRIETARY RIGHTS. The SOFTWARE is proprietary to VENTANA and all copyright, patent, trade secret, trademark and other intellectual property rights are and shall remain the valuable property of VENTANA. You may not reverse engineer, decompile, or disassemble the SOFTWARE. You agree to take all necessary steps to ensure that the provisions of this License are not violated by you or by any person under your control or in your service.

Vensim PLE User's Guide

5. REPRODUCTION AND COPYRIGHT. The SOFTWARE is protected by United States copyright laws and international treaty provisions. You may copy and freely distribute the Vensim PLE Installation program so long as you do not remove or omit any license, copyright or proprietary information or notices that are part of that program. You may not copy, allow anyone else to copy, or otherwise reproduce any part of the SOFTWARE in any other form without the prior written consent of VENTANA, except that you may (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) copy the SOFTWARE to your office computer and your home or portable computer provided that you comply with the restriction on simultaneous use. You may copy the written material accompanying the SOFTWARE for personal use or for educational use as defined in Paragraph 1. You may make a single copy of the Vensim PLE documentation for commercial use with a licensed copy of the Vensim PLE software.

6. RESTRICTIONS ON REDISTRIBUTION. You may not publicly offer for sale the SOFTWARE or any of the written material accompanying the SOFTWARE without the prior written consent of VENTANA. You may not include the SOFTWARE or the written material accompanying the SOFTWARE in any published work without the prior written consent of Ventana Systems, Inc.

7. GENERAL. This license will be governed by the laws of the Commonwealth of Massachusetts.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS LICENSE AND UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS; YOU FURTHER AGREE THAT IT IS THE COMPLETE STATEMENT OF THE AGREEMENT BETWEEN US WHICH SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US, RELATING TO THE SUBJECT MATTER OF THIS LICENSE.

Vensim PLE and PLE Plus Support

Ventana Systems provides email support for questions relating to the use of the Vensim PLE and PLE Plus software. Send questions or problems to **plesupport@vensim.com**.

NOTE Telephone support is not available for Vensim PLE or PLE Plus.

Ventana Systems offers periodic workshops and training on the use of Vensim, as well as a number of other Vensim configurations. If you would like information on these please contact us at:

Ventana Systems, Inc.	US Toll Free: 800 836 7461
60 Jacob Gates Road	International: 617 489 5249
Harvard MA 01451	Fax: 617 489 5316
USA	Email: vensim@vensim.com

or visit our web site at **<http://www.vensim.com>**.