3.4 System Dynamics Tool: *Vensim* Tutorial 2

Introduction to Computational Science: Modeling and Simulation for the Sciences

Angela B. Shiflet and George W. Shiflet
Wofford College

Prerequisite: "*Vensim* Tutorial 1"

Download

Download from the text's website the file *unconstrained.mdl*, which contains a *Vensim* model to accompany this tutorial.

Introduction

This tutorial introduces the following functions and concepts, which subsequent modules employ: Built-in functions and constants, such as *IF THEN ELSE*, *ABS*, *INITIAL*, *EXP*, *TIME*, *PULSE TRAIN*, *TIME STEP*, and *SIN*; relational and logical operators; comparative graphs; and graphical input. Optionally, we cover conveyors, which are useful for some of the later projects.

To understand the material of this tutorial sufficiently, we recommend that you do everything that is requested. While working through the tutorial, answer Quick Review Questions in a separate document.

Built-ins

After clicking with the equation tool () and a stock, flow, or converter, we can enter an equation into that component of a *Vensim* model. Clicking on the ***Functions*** tab towards the middle of the resulting popup menu, we discover a list of built-in functions. The ***More*** tab to the right reveals additional relational and logical operators. In this tutorial, we consider several of these functions and operators that enable us to effectively model many more situations. Table 3.4.1 lists many of the *Vensim* functions, operators, and types along with their formats and meanings. Some of these do not appear as selections in the *Functions* or *More* menus, but we can still type them into equations. Documentation that comes with *Vensim* explains all the functions and features.

Table 3.4.1 Some *Vensim* functions, operators, and types

ABS(*n*)	|*n*|, absolute value *n*
l1 ***:AND:*** *l2*	Logical AND of *l1* and *l2*, where *l1* and *l2* are logical expressions

COS(*r*)	cos(*r*), where *r* is an angle in radians
EXP(*x*)	e^x
IF THEN ELSE (*l*, *s1*, *s2*)	If *l* is true, *s1* is executed; if *l* is false, *s* is returned
Initial(*x*)	Initial value of *x*
INTEGER(*x*)	Integer part (before decimal) of *x*
LOG(*x*, *b*)	$\log_b(x)$, logarithm to the base *b* of *x*
LN(*x*)	ln(*x*), natural logarithm of *x*
MAX(*x1*, *x2*)	Maximum of *x1* and *x2*
MEAN(*x1*, *x2*)	Arithmetic mean of *x1* and *x2*
MIN(*x1*, *x2*)	Minimum of *x1* and *x2*
MODULO(*m*, *n*)	Integer remainder when *m* is divided by *n*
:NOT: *l*	Logical negation of *l*, where *l* is a logical expression
l1 ***:OR:*** *l2*	Logical OR of *l1* and *l2*, where *l1*and *l2* are logical expressions
PULSE TRAIN (*s*, *d*, *i*, *e*)	Pulse of amount 1.0 delivered at time *s* and at every time interval of length *i* afterwards until time *e*; each pulse lasts for duration *d*
SIN(*r*)	sin(*r*), where *r* is an angle in radians
SQRT(*x*)	Square root of *x*
STEP(*h*, *t*)	One-time change of height *h* at time *t*
TAN(*a*)	tan(*a*), where *a* is an angle in radians

Table 3.4.2 Some *Vensim* constants

Final Time	Final time for simulation
Initial Time	Initial time for simulation
Saveper	Length of time between saves of simulated data; should be multiple of *Time Step*
Time	Model simulation's current time
Time Step	Time increment

Initial*, *EXP*,** and ***Time

Open the *Vensim* file *unconstrained.mdl* and save a copy of the file under the name *unconstrainedError.mdl*. Change the dataset name on the main toolbar to *ErrorDS*.

The file models an unconstrained growth situation where the rate of change of the population, *P*, is $dP/dt = 0.1P$ with an initial population of $P_0 = 100$. In Module 3.2 on "Unconstrained Growth," we discovered the following analytical solution to this initial valued differential equation: $P = 100e^{0.10t}$. Suppose we wish to calculate and plot analytical population values along with the simulation population values.

We might want to run the simulation with various initial values of *population* instead of always using 100. Thus, we do not want to type 100 in the equation for analytical population. Fortunately, *Vensim* provides a function, ***Initial***, to return the initial value of a stock, flow, or converter. The software requires that we place the initial value in its own variable. Thus, establish variable (converter) *init_population* and an

arrow (connector) from *population* to this new component. With the equation tool, open *init_population*. After scrolling down to ***Initial*** for ***Type***, click the *Variables* tab (if necessary) and select *population*. When we click *OK*, we find that *Vensim* has dropped the arrow from *population* to *init_population*.

Besides initial population, we also need a dynamic value for time in the analytical formula. Click the **shadow variable tool** () to the right of the rate (flow) tool, click on the model workspace, and select ***Time*** in the popup menu. A component labeled ***<Time>*** in gray tone appears on the diagram.

For the formula, create a variable (converter) with the name *analytical_population* to store the analytical solution for the population, $P = 100e^{0.10t}$, at time *t*. Because the analytically obtained solution uses the initial population, the growth rate, and time, draw arrows (connectors) from *init_population*, *growth_rate*, and <*Time*> to *analytical_population*. With the equation tool, click the latter to enter the equation for $100e^{0.10t}$. After selecting *init_population* and typing the multiplication symbol, *, we enter the *Vensim* equivalent of $e^{0.10t}$. If necessary, click the *Functions* tab. Select the *Vensim* built-in exponential function, ***EXP***, and click the ***Add Sel*** button to insert the function call into the equation. Click on *growth_rate* from the *Variables* menu to place the variable inside the parentheses for *EXP*. The exponent is the product of *growth_rate*, which in this example has a value of 0.10, and the current time, which is the *Vensim* built-in <*Time*>.

Quick Review Question 1 Give the *Vensim* equation for *analytical_population*, which in mathematics is P_0e^{rt}, where P_0 is the *init_population*, *r* is the *growth_rate*, and *t* is *Time*.

ABS

Module 2.2 on "Errors" defines relative error as $\frac{|correct - result|}{|correct|}$. To have *Vensim* calculate this error of the simulation population at every time step, first make a variable with the name *relative_error* and connect *population* and *analytical_population* to this new converter. Then, after selecting *relative_error* with the equation tool, enter an equation. The *Vensim* built-in ***ABS*** returns the absolute value of an expression. Complete the formula. Run the simulation generating a dataset called *ErrorDS*. Display a graph for *population* and *analytical_population* and a table for *population*, *analytical_population*, and *relative_error*.

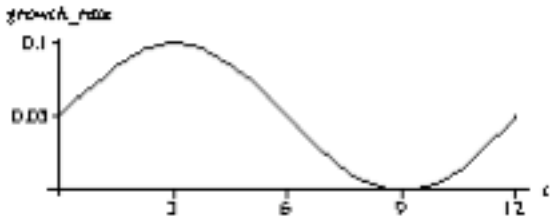
Quick Review Question 2 Give the *Vensim* formula for *relative_error*.

Sine and Cosine

For the next example, save the downloaded file, *unconstrained.mdl*, as *periodic.mdl*. Open the new file, and change the dataset name to *PeriodicDS*.

Suppose we wish to illustrate a periodic growth whose rate is 5% at the beginning of the year, increases to 10% by the beginning of April, is 0% six months later, and returns to 5% with the new year (see Figure 3.4.1). To model such periodicity, we can employ the trigonometric function sine or cosine, which are ***SIN*** and ***COS***, respectively, in *Vensim*.

Figure 3.4.1 Periodic growth rate

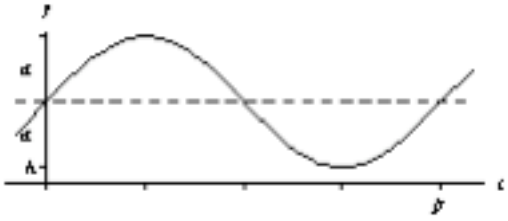


Module 8.2 with a "Function Tutorial" discusses trigonometric functions in greater detail. In general, the graph in Figure 3.4.2 as the formula

$a \sin(2\pi t/p) + h$

where t is the independent variable t; a is the **amplitude**, or height above the horizontal line through the center of the graph; p is the **period**, or length on the horizontal axis before the graph starts to repeat; and h is the lowest height.

Figure 3.4.2 Graph of $a \sin(2\pi t/p) + h$



Click on the converter *growth_rate* with the equation tool and enter the appropriate formula using 3.1416 as an approximation for π. Run the simulation generating a dataset called *PeriodicDS*. Display a graph for *population* and a table for *growth_rate* and *population*.

Quick Review Question 3 Give the equation for *growth_rate* so that its periodic graph has amplitude 0.05, period 12 months, and starts at 0.05 as in Figure 3.4.1.

PULSE TRAIN

For the next example, save the downloaded file, *unconstrained.mdl*, as *pulse.mdl*. Open the new file, and change the dataset name to *PulseDS*. Using *Settings…* of the *Model* menu, change time units to be in Hours instead of Months and the final time to 8 Hours.

With equation panels for the various components, change population to be in bacteria instead of people and time to be in Hours. Perform a *Units Check* from the *Model* menu to verify that the adjustments are consistent.

Suppose the unconstrained growth of a colony of bacteria on a Petri dish is tempered by a researcher removing 50 bacteria every eight hours starting at hour 1. For the model, we make the simplifying assumption that the scientist is able to extract a constant number of bacteria. We can accomplish this task with the *Vensim* function ***PULSE TRAIN***, which has the following format:

PULSE TRAIN(*initial_time*, *width*, *interval*, *end_time*)

From *initial_time* to *end_time*, the function repeatedly returns a pulse value of 1.0 for *width* length of time, where *interval* is the length of time between pulses. If *width* is less than the length of one time step of the simulation (*TIME STEP*), then each pulse lasts one time step. To obtain a pulse other than 1.0, such as 50, we multiply the function call by that value. Thus, for our example, we multiply by 50; *initial_time* is 1; *width* is 0, which forces the length of a pulse to be one time step; *interval* is 8; and *end_time* is 8. An interval value of 0 or greater than the length of the simulation results in a one-time pulse.

In *pulse.mdl*, have a flow called *removal* coming out of *population*. Create three variables (converters) called *amount_removed*, *init_removal_time*, and *frequency_of_removal*; and connect each to the flow *removal*. Enter formulas for *removal* and *population* and values for each of the converters as described in the previous paragraph. Run the simulation generating a dataset called *PulseDS*. Display a graph for *population* and a table for *growth*, *removal*, and *population*.

Quick Review Question 4

- **a.** Give the equation for the flow *removal*.
- **b.** Give the units for *removal*.
- **c.** Give the equation for *population*.

Quick Review Question 5 Without changing *amount_removed* or *init_removal_time*, using the *Vensim* model, determine the largest value (as a multiple of *TIME STEP* = 0.25) of *frequency_of_removal* that will cause the population of bacteria to go to zero eventually, but not necessarily in 8 hours.

Logic

For the next example, save the downloaded file, *unconstrained.mdl*, as *logicIF.mdl*. Open the new file, and change the dataset name to *LogicIFDS*.

Frequently, we want the computer to do one of two things based on a situation. For instance, suppose a population of bacteria has a growth rate of 10% if its size is less than some threshold, such as 1000, but a growth rate of 5% for larger sizes. To model the situation we use *IF THEN ELSE*. The format of the combination of these elements is as follows:

IF THEN ELSE (*condition*, *choice1*, *choice2*)

If logical expression *condition* is true, then the construct returns *choice1*; otherwise, the returned value is *choice2*. Thus, the equation for *growth_rate* described above is as follows:

```
IF THEN ELSE (population < threshold, 0.1, 0.05)
```

Add a variable (converter) for *threshold* and arrows (connectors) from *threshold* and *population* to *growth_rate* in the *Vensim* model. Change the equation for *growth_rate* as described and run the model.

Quick Review Question 6 Describe the appearance of the graph of *population*.

The "less-than" symbol, <, in the condition of the *IF* is an example of a relational operator. A **relational operator** is a symbol that we use to test the relationship between two expressions, such as the two variables *population* and *threshold*. Table 3.4.2 lists the six relational operators in *Vensim*, which are available by typing or with the *More* tab in an equations panel.

Table 3.4.3 *Vensim*'s relational operators

Relational Operator	Meaning
=	equal to
>	greater than
<	less than
<>	not equal to
>=	greater than or equal to
<=	less than or equal to

Definition A **relational operator** is a symbol that we use to test the relationship between two expressions. The relational operators in *Vensim* are **= (equal to), > (greater than), < (less than), <> (not equal to), >= (greater than or equal to), and <= (less than or equal to)**.

Quick Review Question 7 Consider the following equation:

```
IF THEN ELSE (population < threshold, 0.1, 0.05)
```

Keeping *population* and *threshold* in the same order, write an equivalent equation to the expression than employs the >= symbol. Implement your answer in the *Vensim* model.

Logical Operators

For the next example, save the downloaded file, *unconstrained.mdl*, as *logicalAND.mdl*. Open the new file, and change the dataset name to *LogicalANDDS*.

We use **logical operators** to combine or negate expressions containing relational operators. For example, suppose when the number of bacteria is between 500 and 1000, a scientist refrigerates the Petri dish, which results in a lower growth rate (*growth_rate_2* = 5%). However, at room temperature, the growth rate returns to its initial value (*growth_rate_1* = 10%). To write this expression for *growth*, we employ the logical operator ***:AND:*** in conjunction with the relational operators < and >, all of which are available through the *More* tab on an equations panel. For clarity, we are careful to enclose each relational expression in parentheses, as follows:

```
IF THEN ELSE ((500 < population) :AND: (population < 1000),
   growth rate 2 * population, growth rate 1 * population)
```

The **compound condition**, (500 < *population*) *:AND:* (*population* < 1000), is true only when both (500 < *population*) and (*population* < 1000) are both true. In every other circumstance, the condition is false. Table 3.4.3 summarizes this rule in a **truth table** with "T" and "F" indicating true and false, respectively. With *p* representing (500 < *population*) and *q* representing (*population* < 1000), we read the first line of this table as, "When *p* is false and *q* is false, then (*p*) *:AND:* (*q*) is false." Notice that the only way to get a true from an *:AND:* is for both (or all) conditions to be true.

Table 3.4.4 Truth table for (*p*) *:AND:* (*q*)

p	***q***	**(*p*) *:AND:* (*q*)**	**Interpretation**
F	F	**F**	(false) *:AND:* (false) is (false)
F	T	**F**	(false) *:AND:* (true) is (false)
T	F	**F**	(true) *:AND:* (false) is (false)
T	T	**T**	(true) *:AND:* (true) is (true)

In *logicalAND.mdl*, change the name of *growth_rate* to *growth_rate_1*. Add another variable (converter), *growth_rate_2*, with constant value 0.05 and connect it to *growth*. Adjust the equation for *growth* as above to employ the rate *growth_rate_2*, when the population is between 500 and 1000. Run the simulation and observe the effect on the graph and table values.

Quick Review Question 8 In the equation for *growth*, change the condition "(500 < population) :AND: (population < 1000)" to "(500 < population < 1000)", which, as we will see, is incorrect. Click the *Check Syntax* button, and give the error message. Although in mathematics we can have a condition such as 500 < *x* < 1000, in *Vensim* we must use *:AND:* between the two relational expressions. Correct the equation for *growth* and *Check Syntax* again.

When at least one of two conditions must be true in order for a compound condition to be true, we use the logical operator ***:OR:***. For example, the compound condition (*population* <= 500) *:OR:* (1000 <= *population*) is true in every situation, except when both (*population* <= 500) and (1000 <= *population*) are false; that is, when *population* is exclusively between 500 and 1000. Table 3.4.4 has the truth table for (*p*) *:OR:* (*q*). We

read the second line of the table as, "If *p* is false or *q* is true, then (*p*) *:OR: (q*) is true." As that and the remaining lines reveal, if *p* or *q* or both are true, then *p :OR: q* is true.

Table 3.4.5 Truth table for (*p*) *:OR: (q*)

p	***q***	**(*p*) *:OR: (q*)**	**Interpretation**
F	F	**F**	(false) *:OR:* (false) is false
F	T	**T**	(false) *:OR:* (true) is true
T	F	**T**	(true) *:OR:* (false) is true
T	T	**T**	(true) :OR: (true) is true

Quick Review Question 9 Save *logicalAND.mdl* as *logicalOR.mdl*; and open the new file, renaming the dataset *LogicalORDS*. In *logicalOR.mdl*, change the equation for *growth* to have the condition (*population* <= 500) *:OR:* (1000 <= *population*) for the *IF*. Change the remainder of the equation to obtain equivalent results to the above simulation, where the growth rate is 5% for populations between 500 and 1000 and 10% otherwise. Give the *IF THEN ELSE* statement.

A third logical operator, ***:NOT:***, obeys Table 3.4.5. As the table indicates, this operator reverses the truth value of the expression to its immediate right. We can accomplish the same result by changing an expression so that it uses the inverse relational operator. For example,

```
IF THEN ELSE (:NOT:(population < threshold))…
```

is equivalent to

```
IF THEN ELSE (population >= threshold)…
```

In many cases, this latter notation is preferable because it is simpler.

Table 3.4.6 Truth table for *:NOT:*(*p*)

p	***:NOT:*(*p*)**	**Interpretation**
F	**T**	*:NOT:* (false) is true
T	**F**	*:NOT:* (true) is false

Definition A **logical operator** is a symbol that we use to combine or negate expressions that are true or false. The logical operators in *Vensim* are ***:NOT:***, ***:AND:***, and ***:OR:***.

Quick Review Question 10 Save *logicalAND.mdl* as *logicalNOT.mdl*; and open the new file, renaming the dataset *LogicalNOTDS*. In *logicalNOT.mdl*, alter the *growth*

equation to employ one *:NOT:* as indicated with adjustments to the relational operators and the logical operator:

```
IF THEN ELSE (:NOT:((500 ___ population) ___ (population ___ 1000)),
   growth_rate_2 * population,  growth_rate_1 * population)
```

The resulting simulation should produce results equivalent to those of *logicalAND.mdl*.

TIME STEP

For the next example, save the downloaded file, *pulse.mdl*, as *dt.mdl*. Open the new file, and change the dataset name to *DtDS*.

Under the *Model Settings* menu, *Time Bounds* tab, we specify the interval for the time step, ***TIME STEP***. Sometimes it is useful to employ this constant in a model. For example, suppose each time the population of bacteria reaches 200, a scientist harvests 100 of the bacteria for an experiment. In *dt.mdl*, delete the variables connected to *removal* and have an arrow (connector) from *population* to *removal*.

Quick Review Question 11

- **a.** Using *IF THEN ELSE*, give the equation for *removal* that accomplishes the following: If the population is greater than 200, then return 100, else return 0. Make the change and run the simulation.
- **b.** Add columns for *growth* and *removal* in the table. With *TIME_STEP* = 0.25, run the simulation. Give the values for time, *population*, *growth*, and *removal* when the population first exceeds 200.
- **c.** Give the values for time and *population* at the next time step.
- **d.** For the values from Part b, compute *population* + *growth* – *removal*. Does the result equal the population from Part c?
- **e.** As indicated in section "Difference Equation" of Module 3.2 on "Unconstrained Growth," *growth* is multiplied by the time step (*dt*), which is *TIME_STEP* in *Vensim*, before being added to *population*. Similarly, at each time step, *removal* * *TIME_STEP*, not just *removal*, is subtracted from *population*. Give the difference equation for *population*(*t*). Click the document icon (*DOC*), and give the equation for *population*.
- **f.** For the values in Part b, compute the difference equation value for *population*(*t*). Does this result agree with the value of *population* from Part c?
- **g.** Suppose when the population exceeds 200, we wish to remove 100 bacteria, not 25. To cancel out the effect of *Vensim*'s multiplication by *TIME_STEP*, we divide 100 by *TIME_STEP* in the equation for *growth*. Create a shadow variable for *TIME_STEP*, and connect <*TIME_STEP*> to *removal*. Give the resulting *IF THEN ELSE* equation for *removal*. Implement this change and run the simulation, observing the graph and table.
- **h.** Give the values for time, *population*, *growth*, and *removal* when the population first exceeds 200.
- **i.** Give the values for time and *population* at the next time step.

j. For the values in Part h, compute the difference equation value for *population*(*t*). Does this result agree with the value of *population* from Part i?

Comparative Graphs

For the next example, save the downloaded file, *unconstrained.mdl*, as *comparative.mdl*. Open the new file.

Suppose we wish to compare the effect of unconstrained growth on population using various growth rates, such as 0.10, 0.11, 0.12, and 0.13. To do so, run the simulation four times, once for each of the growth rates, and save the datasets under different names, such as *ComparativeDS10*, *ComparativeDS11*, *ComparativeDS12*, and *ComparativeDS13*.

Using the control panel, modify the *Unconstrained_Growth* graph. Have four variables, all of which are *population*, on the same scale. To the right of each variable, give the corresponding dataset, such as *ComparativeDS10*. Enter the title "Populations for Rates 0.10 to 0.13", and click *OK*. Using the control panel, modify the table *Unconstrained_Growth_Table* to display each population. In the options panel, show the line markers (1, 2, 3, and 4) on the graph lines, and display the graph in monochrome with only solid lines. Display the graph and table. The resulting graph and the end of the table are as in Figures 3 and 4, respectively. Comparison of the results reveals the dramatic impact on the population of even a 1% increase in the growth rate.

Figure 3.4.3 Graphs for comparative simulations

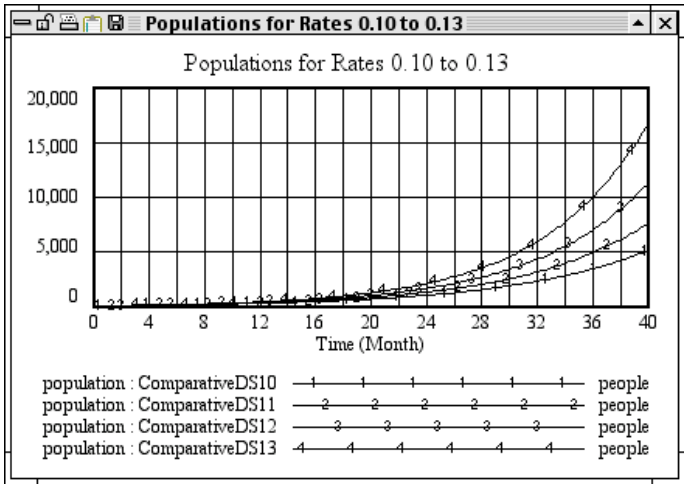


Figure 3.4.4 End of table for comparative simulation

Time (Month)	population	population	population	population
…	…	…	…	...
38.75	4,594	6,701	9,767	14,221
39	4,708	6,885	10,060	14,684
39.25	4,826	7,075	10,362	15,161
39.5	4,947	7,269	10,672	15,654

39.75	5,071	7,469	10,993	16,162
40	5,197	7,675	11,322	16,688

Quick Review Question 12 Lock the current graph and table. Generate a comparative graph and table where (initial) populations are 100, 200, 300, 400, and 500. Give the populations for time = 40 hours.

Graphical Input

For the next example, save the downloaded file, *unconstrained.mdl*, as *graphInput.mdl*. Open the new file, and change the dataset name to *GraphInputDS*.

Sometimes we have a concept of the trend of a variable (converter) or rate (flow) without knowing an expression to represent the equation. For example, perhaps we have experimental data that we wish to use in a model. In this case, we can employ graphical input. Suppose we know that *growth_rate* has a certain shape that depends on the time. In the model, generate a shadow variable for *Time*, and connect it to *growth_rate*. With the equation tool, open the equation panel for *growth_rate*; for *Type*, select *Auxillary*; and in the dropdown box beneath, select *with Lookup*. After clicking the *Variables* tab, click *Time*. Thus, we are asking *Vensim* to look up a value for *growth_rate* from a graph of *growth_rate* versus *Time*. Click the *AsGraph* button to start this graph. We can either enter the raw data in the *Time* (***Input***) and *growth_rate* (***Output***) columns on the left, or we can click on appropriate points in the graph. For example, suppose the growth rate starts at 0.10, decreases to almost 0, and then increases again. Adjust the maximum *Time*, ***X-max***, to be 40 and the maximum *growth_rate* graphical value, ***Y-max***, to be 0.1. By clicking on the graph, enter values for growth rate related to time as in Figure 3.4.5. Using the *Input* and *Output* boxes, adjust the values as necessary to agree with those in the far right column of Figure 3.4.5. Run the simulation. The resulting graph of *population* versus *Time* appears as in Figure 3.4.6.

Figure 3.4.5 Graphical input

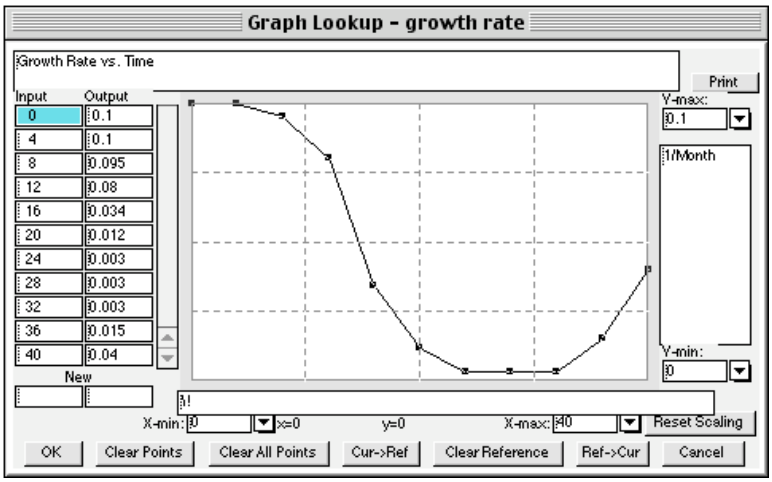
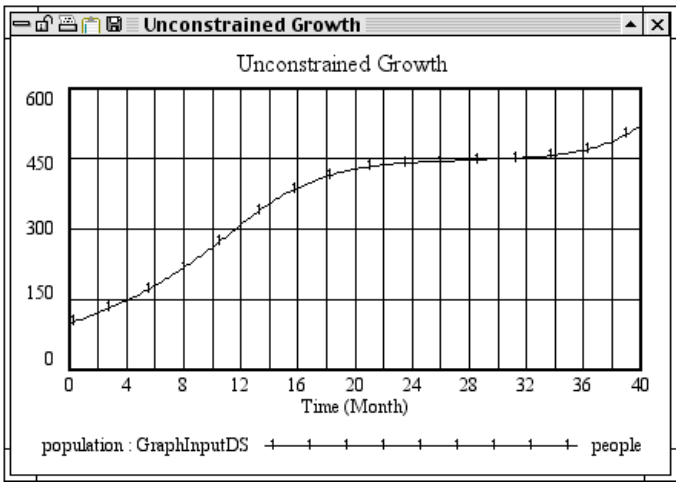


Figure 3.4.6 Resulting graph from *growth_rate* input graph in Figure 3.4.5



Quick Review Question 13 Delete the arrow from <*Time*> to *growth_rate*, and form an arrow connecting *population* to *growth_rate*. Click on *growth_rate* with the equation tool and in the second text box change *WITH LOOKUP* variable from *Time* to *population*. Then, click the *As Graph* button to display the input graph. By clicking the ***Clear All Points*** button on the bottom left, clear the graphical input for *growth_rate*. Generate graphical input for *growth_rate* versus *population*, not time. Have *growth_rate* vary from 0 to 0.1 and *population* from 100 to 10000. Use 10 data points. For a population of size less than 300, have a growth rate of 0.03. For all other populations, have a growth rate of 0.100. Perform the simulation copying over the dataset *GraphInputDS*. Describe the shape of the graph and explain the results.

Conveyor

The material in this section is useful for Project 4 in Module 6.5 on "Modeling Malaria" and could be used for several projects in Module 6.2 on "Spread of SARS" and in Chapter 7.

Sometimes in a model we wish to indicate that each input amount of material remains in that stock (box variable) for a fixed amount of time before exiting. Thus, the stock is a **conveyor** that processes each discrete input batch for a certain amount of time. For example, such a conveyor could model a group of people infected by a virus that has an incubation period of three days.

As another example, with such a conveyor we could model the blood supply at a new blood bank, where processing and screening of a donation takes one week. Start a new model with box variables (stocks) for *processing* and *total_out*. Have a flow (*input*) into *processing* and a flow (*output*) from *processing* to *total_out*. The basic unit of time should be one day. Using *Settings…* on the *Model* menu, let the simulation run for the 12 days with *TIME STEP* = 0.25 days. Make the value for *input* be 100 pints per day and

the initial values of *processing* and *total_out* be 0 pints. Have an arrow (connector) go from *input* to *output*.

Using the equation tool, click on the *output* flow. For its equation, we use the built-in function ***DELAY FIXED*** with the following general form:

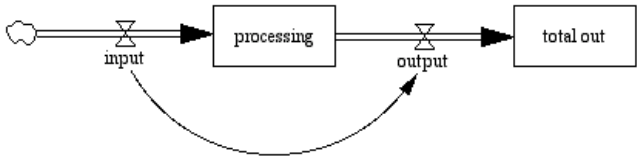
```
DELAY FIXED(amount, duration, start)
```

The statement says, starting at time *start*, delay the amount, *amount*, for a fixed amount of time, *duration*. Thus, we can make the box variable *processing* a conveyor with the following equation for *output:*

```
DELAY FIXED(input, 7 ,0 )
```

From the beginning of the simulation, the transit time from input to output takes 7 days. After clicking *OK*, we see a model diagram similar to Figure 3.4.7 with the *output* flow regulating the conveyor. Save your work in a file called *conveyor.mdl*.

Figure 3.4.7 Model with *output* flow regulating conveyor *processing*



Generate a graph for *processing* and a table containing *processing* and *total_out*. Run the simulation, and save again. We note that *processing* builds steadily for the first 7 days, increasing by 100 pints per day. With *TIME_STEP* being 0.25, *processing* is actually increasing by 25 pints per quarter of a day (i.e., 25 pints each 6 hours) for the first week. During that time, *total_out* remains 0. Then, at 7.25 days, the 25 pints that entered the conveyor, *processing*, at time 0.25 days leave *processing* and go into *total_out*. From then on, the quantity in *processing* is in a steady state with the same number of pints entering as leaving at any time step.

Quick Review Question 14 Give the values of each of the following at time 12 days:

a. *processing*
b. *total_out*

Projects

For additional projects, see Module 7.4 on "Cardiovascular System—A Pressure-Filled Model" and Module 7.8 on "Mercury Pollution—Getting on Our Nerves."

Reference

Kirkwood, Craig W., updated by Jennifer Cihla Vender. 2002. *Vensim® PLE Quick Reference and Tutorial*,Ventana Systems, Inc. http://www.public.asu.edu/~kirkwood/sysdyn/VenPLE.pdf

Vensim 5 Modeling Guide. 2003. Ventana Systems, Inc. http://www.vensim.com/

Vensim 5 Reference Manual. 2003. Ventana Systems, Inc. http://www.vensim.com/

Vensim® Ventana® Simulation Environment, User's Guide Version 5. 2002. Ventana Systems, Inc. http://www.vensim.com/