

3.4 System Dynamics Tool: *Berkeley Madonna Tutorial 2*

*Introduction to Computational Science:
Modeling and Simulation for the Sciences*

Angela B. Shiflet and George W. Shiflet
Wofford College
© 2006 by Princeton University Press

Prerequisite: "Berkeley Madonna Tutorial 1"

Download

Download from the text's website the file *unconstrained*, which contains a *Berkeley Madonna* 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*, *INIT*, *EXP*, *TIME*, *PI*, *PULSE*, *DT*, *SIN*, and *COS*; 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

We can enter equations into a reservoir, flow, or formula of a *Berkeley Madonna* model. **Equation Help** under the **Help** menu summarizes *Berkeley Madonna*'s functions and features. In this tutorial, we consider several of these functions that enable us to effectively model many more situations.

Table 3.4.1 lists many of the *Berkeley Madonna* functions along with their formats and meanings. The following tutorial illustrates a number of these through examples.

Table 3.4.1 Some *Berkeley Madonna* functions

<i>ABS</i> (<i>n</i>)	$ n $, absolute value <i>n</i>
<i>l1 AND l2</i>	Logical AND of <i>l1</i> and <i>l2</i> , where <i>l1</i> and <i>l2</i> are logical expressions
<i>COS</i> (<i>r</i>)	$\cos(r)$, where <i>r</i> is an angle in radians
<i>DT</i>	Time increment
<i>ELSE</i> (<i>s2</i>)	In <i>IF l THEN s1 ELSE s2</i> , if <i>l</i> is false, <i>s2</i> is returned
<i>EXP</i> (<i>x</i>)	e^x

IF	In <i>IF l THEN s1 ELSE s2</i> , if <i>l</i> is true, <i>s1</i> is executed; if <i>l</i> is false, <i>s</i> is returned
INIT(x)	Initial value of <i>x</i>
INT(x)	Largest integer less than or equal to <i>x</i>
LOG10(x)	$\log_{10}(x)$, logarithm to the base 10 of <i>x</i> ; common logarithm of <i>x</i>
LOGN(x)	$\ln(x)$, natural logarithm of <i>x</i>
MAX(x1, x2, ...)	Maximum of <i>x1</i> , <i>x2</i> , ...
MEAN(x1, x2, ...)	Arithmetic mean of <i>x1</i> , <i>x2</i> , ...
MIN(x1, x2, ...)	Minimum of <i>x1</i> , <i>x2</i> , ...
MOD(m, n)	Integer remainder when <i>m</i> is divided by <i>n</i>
NOT l	Logical negation of <i>l</i> , where <i>l</i> is a logical expression
l1 OR l2	Logical OR of <i>l1</i> and <i>l2</i> , where <i>l1</i> and <i>l2</i> are logical expressions
PI	Approximation of $\pi = 3.14159\dots$
PULSE(a, t, i)	Pulse of amount <i>a</i> first delivered at time <i>t</i> and at every time interval of length <i>i</i> afterwards
ROUND(x)	<i>x</i> rounded to the nearest integer
SIN(r)	$\sin(r)$, where <i>r</i> is an angle in radians
SQRT(x)	Square root of <i>x</i>
STEP(h, t)	0 before time <i>t</i> and <i>h</i> for time $\geq t$
TAN(a)	$\tan(a)$, where <i>a</i> is an angle in radians
THEN	In <i>IF l THEN s1 ELSE s2</i> , if <i>l</i> is true, <i>s1</i> is executed
TIME	Model simulation's current time

INIT, EXP, and TIME

Open the *Berkeley Madonna* file *unconstrained* and save a copy of the file under the name *unconstrainedError*.

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. If necessary, from the **Flowchart menu**, select **Show Flowchart**.

In the flowchart, create a formula 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 and the growth rate, draw arcs from the reservoir *population* and the formula *growth_rate* to the new formula, *analytical_population*. Double-click the latter to enter the equation for $100e^{0.10t}$. 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, *Berkeley Madonna* provides a function, **INIT**, to return the initial value of a reservoir, flow, or formula. The software is not case sensitive, so we can use **INIT** or *init*. After typing the name of the function and a left parenthesis, double-click on *population* in the **Required Inputs** menu and type a right parenthesis to obtain **INIT(population)**. After a multiplication symbol, *, we enter the *Berkeley Madonna* equivalent of $e^{0.10t}$. **EXP** is the *Berkeley Madonna* built-in exponential

function. Double-click on *growth_rate* from the *Required Inputs* 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 *Berkeley Madonna* built-in *TIME*.

Quick Review Question 1 Give the *Berkeley Madonna* equation for *analytical_population*, which in mathematics is P_0e^{rt} , where P_0 is the initial population, r is the *growth_rate*, and t is the time.

ABS

Module 2.2 on "Errors" defines relative error as $|correct - result|/|correct|$. To have *Berkeley Madonna* calculate this error of the simulation population at every time step, first make a flowchart formula with the name *relative_error* and connect *population* and *analytical_population* to this new formula. Then, double-click on the latter to enter an equation. The *Berkeley Madonna* built-in *ABS* returns the absolute value of an expression. Complete the formula. Run the simulation generating a graph for *population* and *analytical_population* and a table for *population*, *analytical_population*, and *relative_error*.

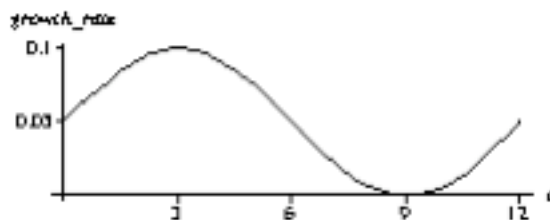
Quick Review Question 2 Give the *Berkeley Madonna* formula for *relative_error*.

Sine and Cosine

For the next example, save the downloaded file, *unconstrained*, as *periodic*, and open the new file.

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 *Berkeley Madonna*.

Figure 3.4.1 Periodic growth rate

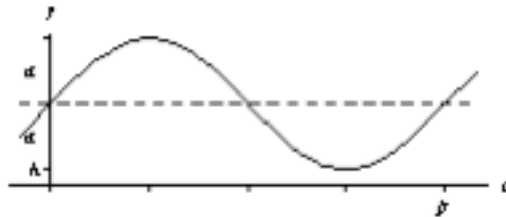


Module 8.2 with a "Function Tutorial" discusses trigonometric functions in greater detail. In general, the graph in Figure 3.4.2 as the formula

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

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



Double-click on the formula *growth_rate* and enter the appropriate formula to obtain the graph for Figure 3.4.1. Run the simulation generating 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

For the next example, save the downloaded file, *unconstrained*, as *pulse*, and open the new file.

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 *Berkeley Madonna* function *PULSE*, which has the following format:

PULSE(*amount*, *initial_time*, *interval*)

where *amount* is the amount that the function returns during a pulse, *initial_time* is the time of the first pulse, and *interval* is the length of time between pulses. Thus, for our example, *amount* is 50; *initial_time* is 1; and *interval* is 8. An interval value greater than the length of the simulation results in a one-time pulse.

In *pulse*, have a flow called *removal* coming out of *population*. Create three formulas called *amount_removed*, *init_removal_time*, and *frequency_of_removal*; and connect each to the flow *removal*. Enter a formula for *removal* and values for each of the formulas as described in the previous paragraph. Run the simulation.

Quick Review Question 4 Give the equation for the flow *removal*.

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

Logic

For the next example, save the downloaded file, *unconstrained*, as *logicIF*, and open the new file.

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 *condition* **THEN** *choice1* **ELSE** *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 (population < threshold) **THEN** 0.1 **ELSE** 0.05

Add a formula for *threshold* and arcs from *threshold* and *population* to *growth_rate* in the *Berkeley Madonna* 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 *Berkeley Madonna*.

Table 3.4.2 *Berkeley Madonna'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 *Berkeley Madonna* 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 (population < threshold) THEN 0.1 ELSE 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 *Berkeley Madonna* model.

Logical Operators

For the next example, save the downloaded file, *unconstrained*, as *logicalAND*, and open the new file.

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, the 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 >, as follows:

```
IF (500 < population) AND (population < 1000)
THEN growth_rate_2 * population
ELSE growth_rate_1 * population
```

The **compound condition**, $(500 < population) \text{ AND } (population < 1000)$, 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 3.4.3 Truth table for *p AND q*

<i>p</i>	<i>q</i>	<i>p AND q</i>	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*, change the name of *growth_rate* to *growth_rate_1*. Add another formula, *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. Run the simulation. By observing the values in the table, determine which growth rate, *growth_rate_1* = 0.1 or *growth_rate_2* = 0.05, *Berkeley Madonna* is using. Although in mathematics we can have a condition such as $500 < x < 1000$, in *Berkeley Madonna* we must use *AND* between the two relational expressions. Correct the equation for *growth*.

When at least one of two conditions must be true in order for the 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 3.4.4 Truth table for *p OR q*

<i>p</i>	<i>q</i>	<i>p OR q</i>	Interpretation
F	F	F	false <i>OR</i> false is false
F	T	T	false <i>OR</i> true is true
T	F	T	true <i>OR</i> false is true
T	T	T	true <i>OR</i> true is true

Quick Review Question 9 Save *logicalAND* as *logicalOR*, and open the new file. In *logicalOR*, 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 (NOT(population < threshold))
```

is equivalent to

```
IF (population >= threshold)
```

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

Table 3.4.5 Truth table for *NOT p*

<i>p</i>	<i>NOT p</i>	Interpretation
F	T	<i>NOT</i> false is true
T	F	<i>NOT</i> 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 *Berkeley Madonna* are *NOT*, *AND*, and *OR*.

Quick Review Question 10 Save *logicalAND* as *logicalNOT*, and open the new file. In *logicalNOT*, alter the *growth* equation to employ one *NOT* as indicated with adjustments to the relational operators and the logical operator:

```
IF NOT ((500 _____ population) _____ (population _____ 1000))
THEN growth_rate_2 * population
ELSE growth_rate_1 * population
```

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

DT

For the next example, save the file *pulse* as *dt*, and open this new file.

In the parameters window, we specify the interval for the time step, *DT*. 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 file *dt*, delete the formulas connected to *removal* and have an arc from *population* to *removal*.

Quick Review Question 11

- 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.
- Have columns for *population*, *growth*, and *removal* in the table. With *DT* = 0.25, run the simulation. Give the values for time, *population*, *growth*, and *removal* when the population first exceeds 200.
- 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 DT before being added to $population$. Similarly, at each time step, $removal * DT$, not just $removal$, is subtracted from $population$. Give the difference equation for $population(t)$.
- f. For the values in Part b, compute $population(t)$. 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 *Berkeley Madonna's* multiplication by DT , we divide 100 by DT in the equation for $growth$. Give the resulting *IF-THEN-ELSE* equation. 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 $population(t)$. Does this result agree with the value of $population$ from Part i?

Comparative Graphs

For the next example, save the downloaded file, *unconstrained*, as *comparative*, and open this 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. From the *Parameters* menu, select **Batch Runs...** or use the indicated shortcut. From the dropdown menu as in Figure 3.4.3, select parameter $growth_rate$ and indicate 4 runs with initial value 0.10 and final value 0.13. *Berkeley Madonna* computes the values for $growth_rate$ as 0.10, 0.11, 0.12, and 0.13. Click *OK*. Indicate to show labels and parameters on the graph. The resulting graph and the end of the table are as in Figure 3.4.4 and Table 3.4.6, 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 Batch-runs dropdown menu

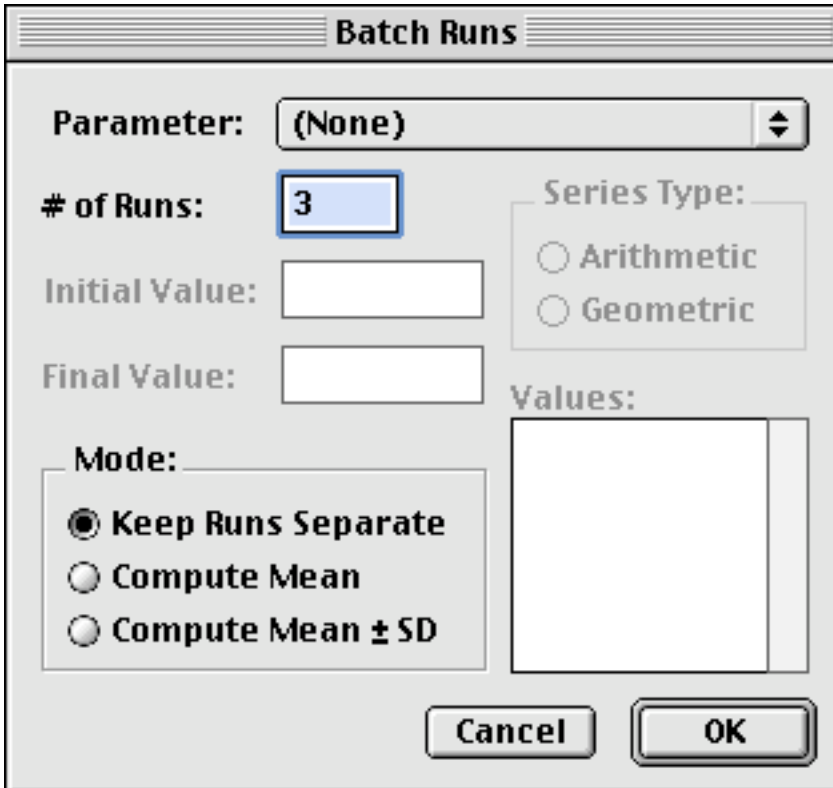


Figure 3.4.4 Graph for comparative simulation

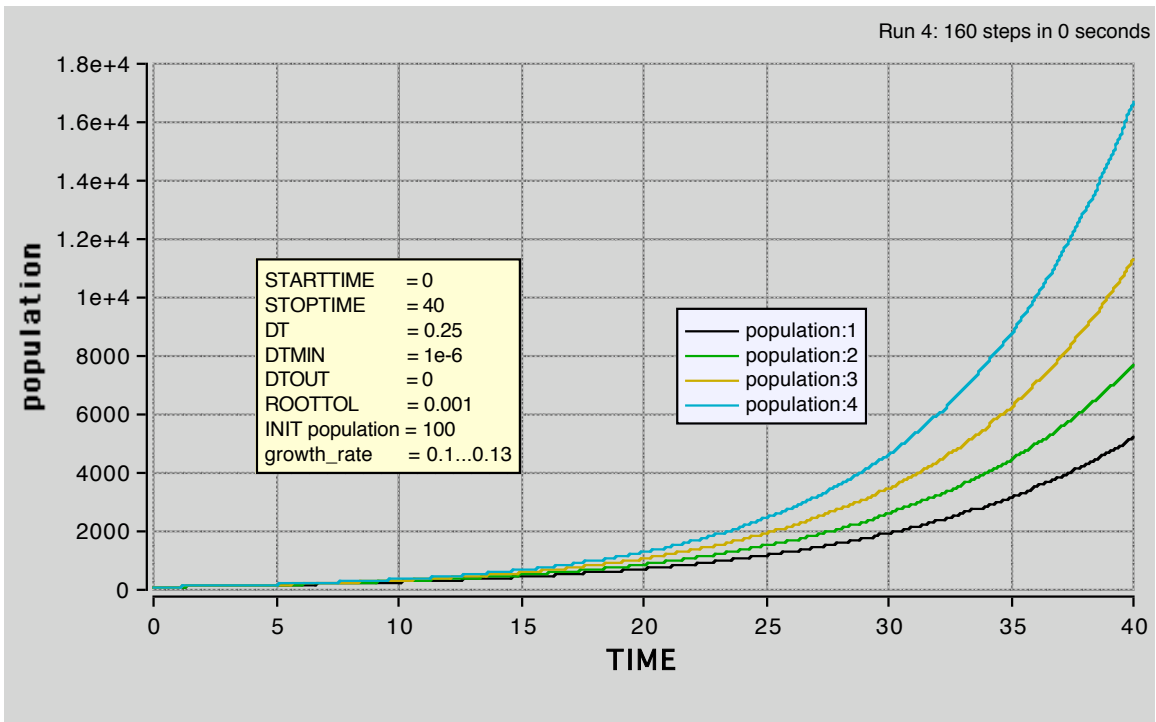



Table 3.4.6 End of table for comparative simulation

Time	population: 1	population: 2	population: 3	population: 4
...
38.75	4,594.09	6,701.68	9,767.19	14,222.0
39.00	4,708.94	6,885.98	10,060.2	14,684.2
39.25	4,826.66	7,075.34	10,362.0	15,161.4
39.50	4,947.33	7,269.91	10,672.9	15,654.1
39.75	5,071.01	7,469.84	10,993.1	16,162.9
40.00	5,197.79	7,675.26	11,322.9	16,688.2

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

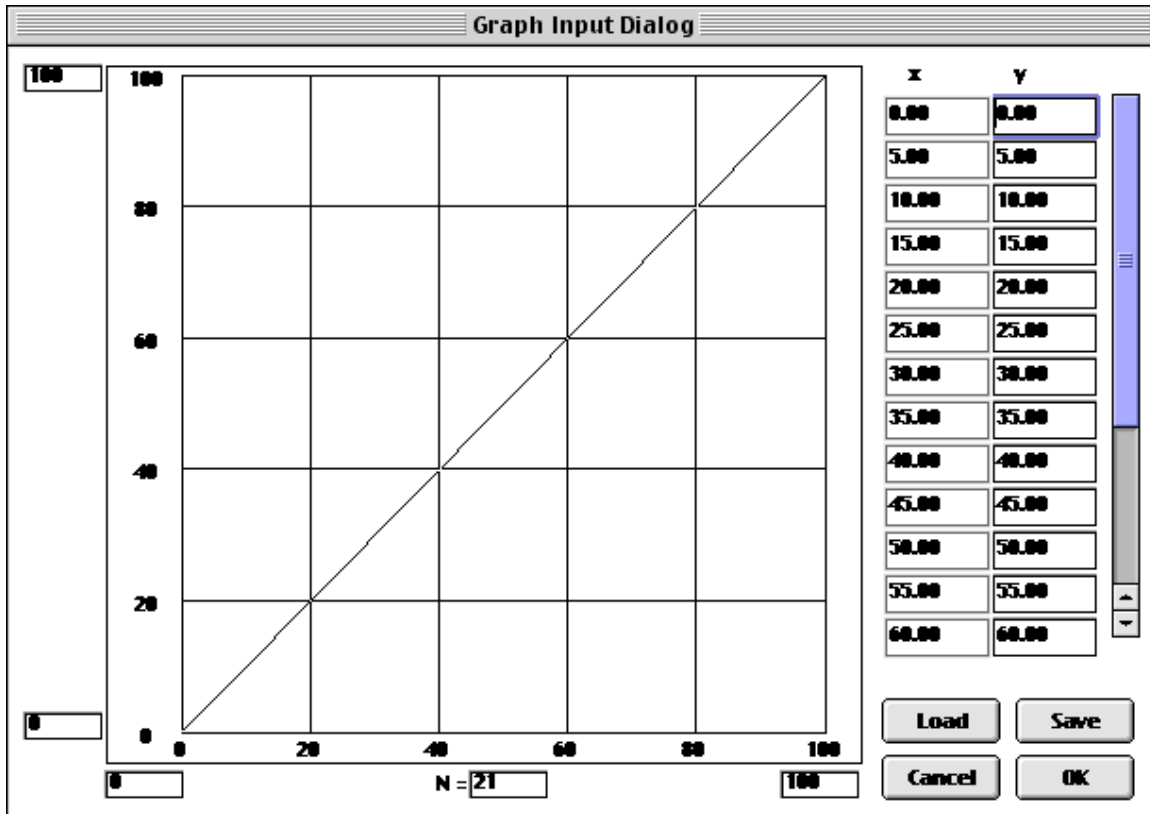
An alternative way to display plots for different simulations together is to click the **Overlay Plots icon**  toward the top left of a graph window. Until deselected, all plots appear in the same graph window. However, we can from the *Graph* menu **Discard Last Run** or **Discard All Runs**.

Graphical Input

For the next example, save the downloaded file, *unconstrained*, as *graphInput*, and open this new file.

Sometimes we have a concept of the trend of a formula or 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. Create another formula component called *growth_rate_times_1000* that connects to *growth_rate*. Change the equation of *growth_rate* to be $0.001 * \text{growth_rate_times_1000}$, so that *growth_rate* is one-thousandth of *growth_rate_times_1000*. Double-click *growth_rate_times_1000*; in place of the equation type *TIME*, the independent variable; and click *Create Graph* button to the right of the pop-up menu. Figure 3.4.5 shows the resulting **graph input dialog window**.

Figure 3.4.5 Default graph input dialog window



The x values represent *TIME* and the y values *growth_rate_times_1000*. We can click and drag on the line to trace a desired graph. For example, suppose the growth rate starts close to 0.1, decreases to almost 0, and then increases again. Thus, *growth_rate_times_1000*, which is 1000 times the growth rate, should start at about 100. By clicking and dragging on the graph, enter values for *growth_rate_times_1000* related to time similar to Figure 3.4.6. Because we are running the simulation only to time 40, values for later times are irrelevant for this example. By clicking *OK*, get out of the graph input window and then the equation window for *growth_rate_times_1000*. Run the simulation. The resulting graphs of *population* and *growth_rate* versus *time* should appear similar to Figure 3.4.7. Save your work.

Figure 3.4.6 Graph input dialog window for *growth_rate_times_1000*

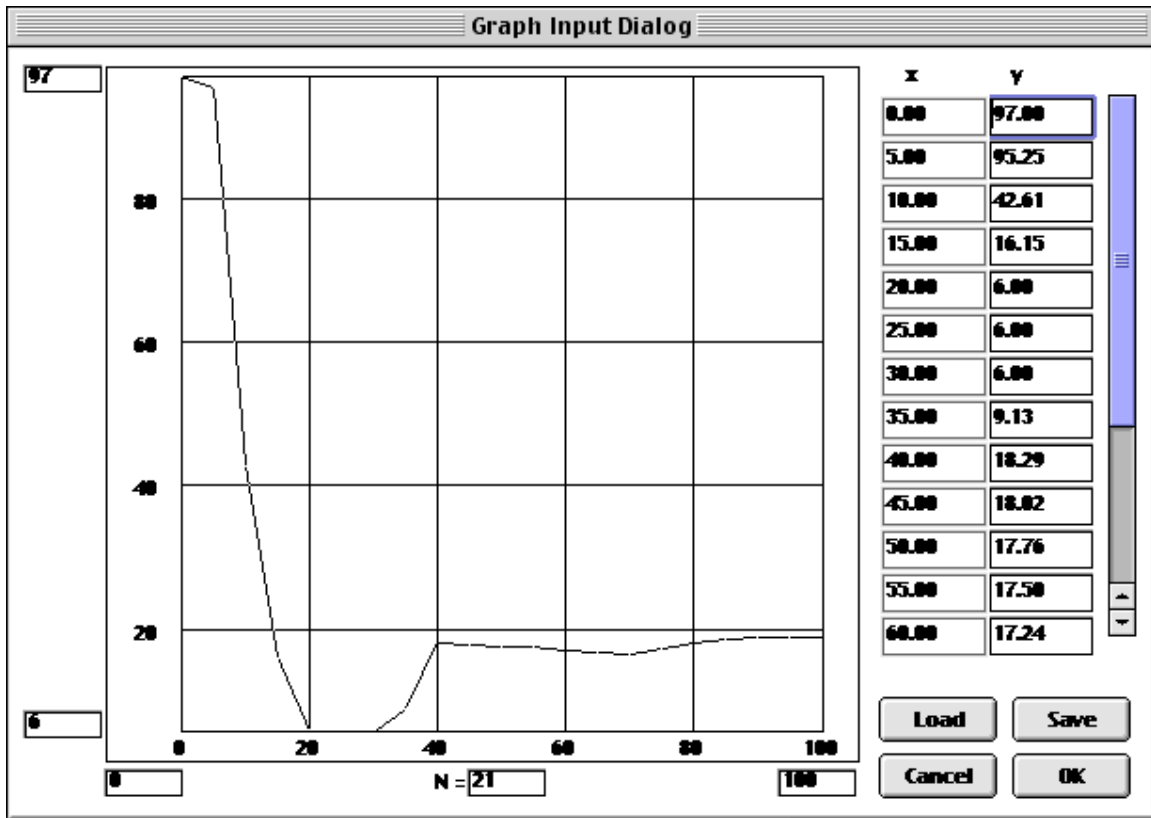
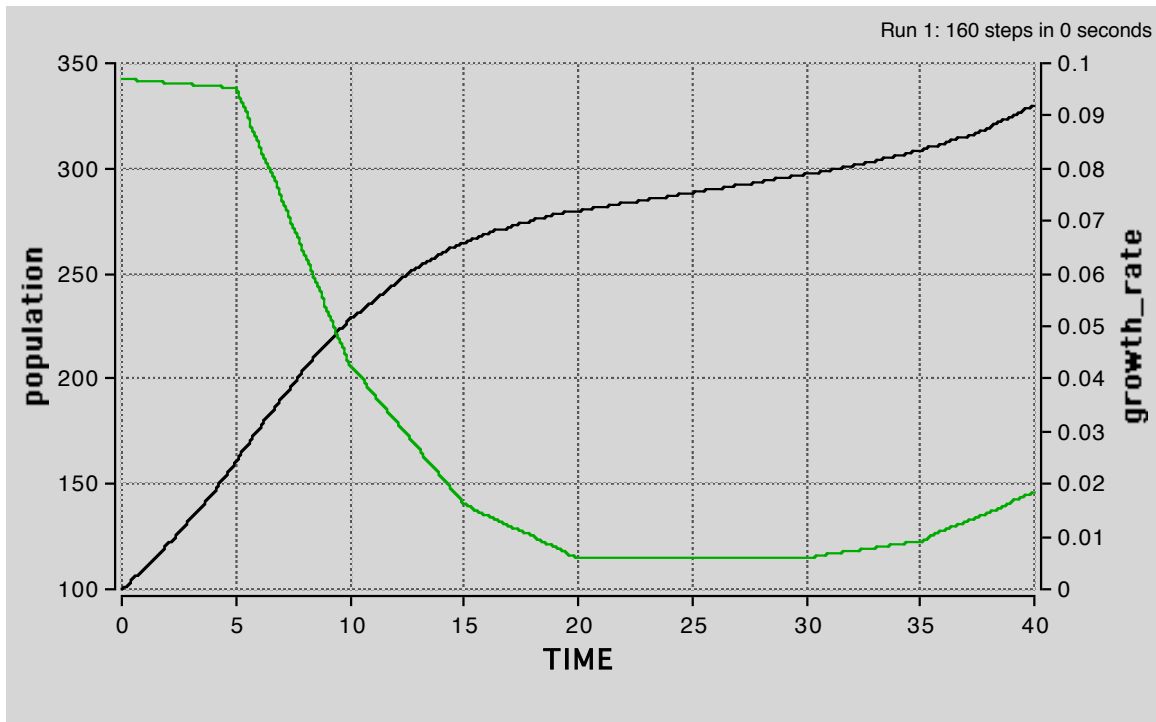


Figure 3.4.7 Resulting graphs from *growth_rate_times_1000* input graph in Figure 3.4.6



Quick Review Question 13 Save the model *graphInput* as *graphInputPop*. Rename *growth_rate_times_1000* as *growth_rate_times_100*, and change the equation for *growth_rate* accordingly. Add another formula component, *population_div_100*, which is *population* divided by 100. Have arcs from *population* to *population_div_100* and from *population_div_100* to *growth_rate_times_100*. Double-click *growth_rate_times_100*.

- What do we type in the equation box for generating graphical input for *growth_rate_times_100* versus *population*, not time?
- Click the *Delete Graph* button. What do we do to view the graph input dialog window?
- With *growth_rate_times_100* varying from 0 to 100, between what values can *growth_rate* vary?
- With *population_div_100* varying from 0 to 100, between what values can *population* vary?
- For *population_div_100* sizes less than 25, corresponding to populations less than 2500, have *growth_rate_times_100* be 20, or a growth rate of 20%. For *population_div_100* values from 25 to 65, have *growth_rate_times_100* decrease linearly from 20 to 0. For *population_div_100* values greater than 65, have *growth_rate_times_100* be 0. Describe the shape of the *population* graph and explain the results.
- Describe the shape of the *growth_rate* graph and explain the results. Save your work.

Save the model *graphInput* as *graphInputNoFlowchart*.

For more precise input with greater options, we can type coordinates of the graph into the equation window. From the *Model* menu, select **Discard Flowchart** so that we can edit equations from that window. If not visible, open the equations window. Note that *Berkeley Madonna* represented our graphical input for *growth_rate_times_1000* as **graph(TIME)** followed by a list of 21 coordinates. Delete "0.001 * *growth_rate_times_1000*" and "*growth_rate_times_1000* =" so that we assign **graph(TIME)** to *growth_rate*. Change the coordinates to be as follows: (0, 0.1), (4, 0.1), (8, 0.095), (12, 0.08), (16, 0.034), (20, 0.012), (24, 0.003), (28, 0.003), (32, 0.003), (36, 0.015), (40, 0.04). Notice that with text input, we can eliminate *growth_rate_times_1000* and can have coordinate values for *TIME* other than multiples of 5. Run the simulation.

Quick Review Question 14 Give the maximum population and when it occurs.

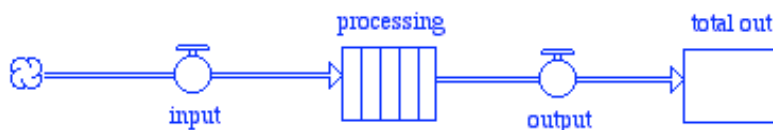
Conveyor

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 reservoir for a fixed amount of time before exiting. Thus, the reservoir 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. Suppose *input* consists of 100 pints per day to the blood bank and the transit time for the conveyor *processing* is 7 days. Output flows from *processing*, through *output*, to reservoir *total_out*. A diagram for such a configuration in the system dynamics tool *STELLA* appears in Figure 3.4.8.

Figure 3.4.8 *STELLA* model with conveyor *processing*



Berkeley Madonna does not have a flowchart component for a conveyor. However, we can enter **discrete equations**, which the software can use to perform a simulation. Start a new model, and save your work in a file called *conveyor*. In the equations window, type equations similar to those that *Berkeley Madonna* previously automatically

generated. We begin by specifying the integration method, start and stop times, and step size, as follows:

```
METHOD EULER
STARTTIME = 0
STOPTIME = 12
DT = 0.25
```

Assigning *CONVEYOR(input_amount, transit_time)* to a conveyor, such as *processing*, indicates that every unit of time *input_amount* enters the conveyor and remains in the conveyor for *transit_time* units of time. The value of an output flow, such as *output*, is *OUTFLOW* of the conveyor. Also, type the following segment of discrete equations to complete the simulation code:

```
input = 100
processing = CONVEYOR(input, 7)
output = OUTFLOW(processing)
INIT total_out = 0
d/dt total_out = + output
```

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 *DT* 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 15 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

Macey, Robert, George Oster, and Tim Zahnley, 2000. *Berkeley Madonna User's Guide*, Version 5.0, University of California. <http://www.berkeleymadonna.com>