3.4 System Dynamics Tool: *STELLA* Version 9 Tutorial 2

Introduction to Computational Science: Modeling and Simulation for the Sciences

Angela B. Shiflet and George W. Shiflet
Wofford College

Prerequisite: "*STELLA* Tutorial 1"

Download

Download from the text's website the file *unconstrained.stm*, which contains a *STELLA* 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*, and *SINWAVE*; 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

In the equation mode, we can enter equations into a stock, flow, or converter of a *STELLA* model. The resulting pop-up menu contains a list of a many of built-in functions that fall into ten categories: array, cycle-time, discrete, financial, logical, mathematical, special purpose, statistical, test input, and trigonometric. In this tutorial, we consider several of these functions that enable us to effectively model many more situations. Browser-accessible documentation that comes with *STELLA* explains all the functions and features.

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

Table 3.4.1 Some *STELLA* functions

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

COSWAVE (a, p)	Time-dependent cosine function of amplitude a and period p
DT	Time increment
ELSE $(s2)$	In *IF*(l) *THEN*$(s1)$ *ELSE*$(s2)$, if l is false, $s2$ is returned
EXP(x)	e^x
FV(r, n, a, p)	Future value of present value (p) with n payments of amount a and interest rate of r per period
IF	In *IF*(l) *THEN*$(s1)$ *ELSE*$(s2)$, if l is true, $s1$ is executed; if l is false, s is returned
INIT(x)	Initial value of x
INT(x)	Largest integer less than or equal to x
LOG10(x)	$\log_{10}(x)$, logarithm to the base 10 of x; common logarithm of x
LOGN(x)	$\log_e(x)$, logarithm to the base e of x; $\ln(x)$, natural logarithm of x
MAX$(x1, x2, \ldots)$	Maximum of $x1, x2, \ldots$; use Euler's method
MEAN$(x1, x2, \ldots)$	Arithmetic mean of $x1, x2, \ldots$
MIN$(x1, x2, \ldots)$	Minimum of $x1, x2, \ldots$; use Euler's method
MOD(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
PI	Approximation of $\pi = 3.14159\ldots$
PMT(r, n, p, f)	Payment every period to go from present value (p) to future value (f) in n payments with interest rate of r per period
PULSE(a, t, i)	Pulse of amount a first delivered at time t and at every time interval of length i afterwards; by default $t = 0$ and $i = DT$
PV(r, n, a, f)	Present value of future value (f) with n payments of amount a and interest rate of r per period
ROUND(x)	x rounded to the nearest integer; use Runge-Kutta
SIN(r)	$\sin(r)$, where r is an angle in radians
SINWAVE(a, p)	Time-dependent sine function of amplitude a and period p
SQRT(x)	Square root of x
STEP(h, t)	0 before time t and h for time $\geq t$
TAN(a)	$\tan(a)$, where a is an angle in radians
THEN	In *IF*(l) *THEN*$(s1)$ *ELSE*$(s2)$, if l is true, $s1$ is executed
TIME	Model simulation's current time

INIT*, *EXP*,** and ***TIME

Open the *STELLA* file *unconstrained.stm* and save a copy of the file under the name *unconstrainedError.stm*.

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. Navigate to the *Model* level in the *STELLA* file.

Create a 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 and the growth rate, draw connectors from the stock *population* and the converter *growth_rate* to the new converter, *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, *STELLA* provides a function, ***INIT***, to return the initial value of a stock, flow, or converter. Under the *Built-ins* menu on the right, scroll down and select *INIT*. With the cursor automatically inside the parentheses, click on *population* in the *Required Inputs* menu to obtain *INIT*(*population*). After typing the multiplication symbol, *, we enter the *STELLA* equivalent of $e^{0.10t}$. Select the *STELLA* built-in exponential function, ***EXP***. 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 *STELLA* built-in ***TIME***.

Quick Review Question 1 Give the *STELLA* 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 *STELLA* calculate this error of the simulation population at every time step, first make a converter with the name *relative_error* and connect *population* and *analytical_population* to this new converter. Then, double-click on the latter to enter an equation. The *STELLA* 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 *STELLA* formula for *relative_error*.

Sine and Cosine

For the next example, save the downloaded file, *unconstrained.stm*, as *periodic.stm*, 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 *STELLA*. However, *STELLA* offers even easier-to-use functions for this example, ***SINWAVE*** or ***COSWAVE***. For these functions we specify the amplitude, or height above the horizontal line through the center of the graph, and the period, or length on the horizontal axis before the graph starts to repeat. In this example, the amplitude is 0.05 and the period is 12 months, as in Figure 3.4.2. As with *SIN*, the graph of *SINWAVE* begins at the origin, while the graphs of *COS* and *COSWAVE* begin at the high point. Thus, we employ *SINWAVE*(0.05, 12) to generate the graph in Figure 3.4.2. However, to obtain the desired graph for Figure 3.4.1, add 0.05 to the result. Double-click on the converter *growth_rate* and enter the appropriate formula. 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 the 0.05 as in Figure 3.4.1.

Figure 3.4.1 Periodic growth rate

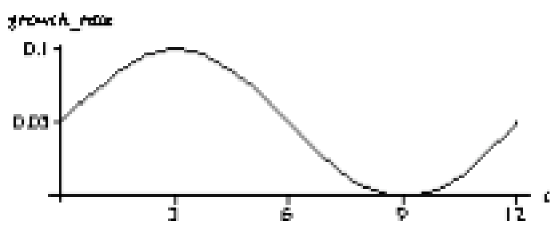
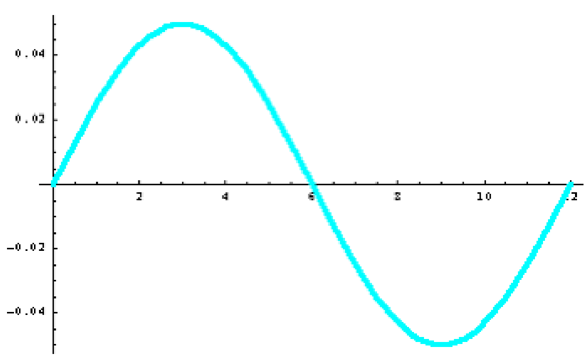


Figure 3.4.2 *SINWAVE*(0.05, 12)



PULSE

For the next example, save the downloaded file, *unconstrained.stm*, as *pulse.stm*, and open this 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 *STELLA* 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 of 0 or greater than the length of the simulation results in a one-time pulse. If we omit *initial_time* and *interval*, such as with *PULSE*(50), the system uses default values of *initial_time* = 0 and *interval* = *DT* so that the pulse occurs every time step from the beginning of the simulation.

In *pulse.stm*, have a flow called *removal* coming out of *population*. Create three converters 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 converters 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 *STELLA* 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.

Logic

For the next example, save the downloaded file, *unconstrained.stm*, as *logicIF.stm*, and open this 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 converter for *threshold* and connectors from *threshold* and *population* to *growth_rate* in the *STELLA* 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 *STELLA*.

Table 3.4.2 *STELLA*'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 *STELLA* 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 *STELLA* model.

Logical Operators

For the next example, save the downloaded file, *unconstrained.stm*, as *logicalAND.stm*, 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 >, being careful to enclose each relational expression in parentheses, as follows:

```
IF (500 < population) AND (population < 1000)
THEN growth_rate_2 * population
ELSE 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 3.4.3 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.stm*, change the name of *growth_rate* to *growth_rate_1*. Add another 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. 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, *STELLA* is using. Although in mathematics we can have a condition such as 500 < *x* < 1000, in *STELLA* 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*)

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.stm* as *logicalOR.stm*, and open the new file. In *logicalOR.stm*, 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*)

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 *STELLA* are ***NOT***, ***AND***, and ***OR***.

Quick Review Question 10 Save *logicalAND.stm* as *logicalNOT.stm*, and open the new file. In *logicalNOT.stm*, 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.stm*.

DT

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

In *Run Specs*, 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 *dt.stm*, delete the converters connected to *removal* and have a connector from *population* to *removal*.

Quick Review Question 11

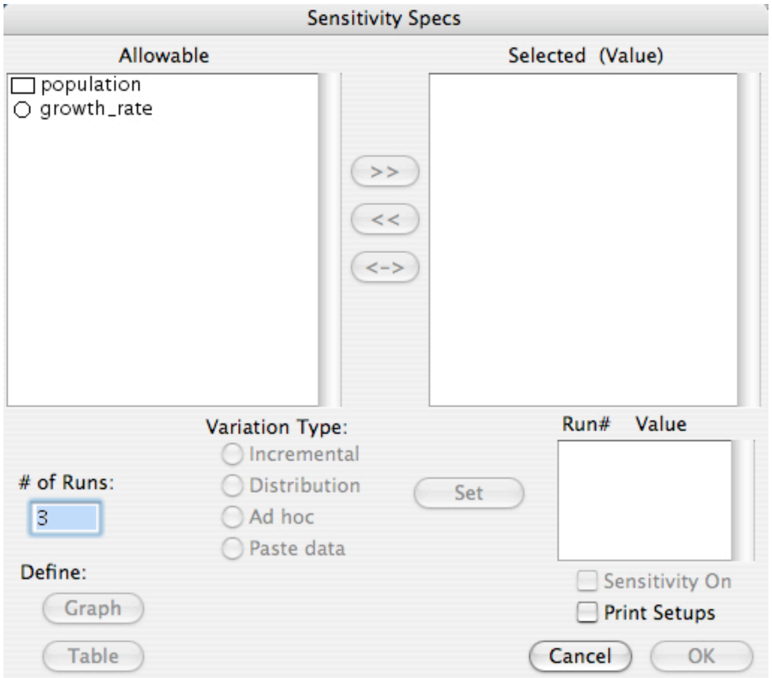
- **a.** Using *IF-THEN-ELSE*, give the equation for *removal* that accomplishes the following: If the population is greater than 200, then return 100, else return 0.
- **b.** Add columns for *growth* and *removal* in the table. With *DT* = 0.25, run the simulation. Give the values for time, *population*, *growth*, and *removal* when the population first exceeds 200.
- **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 *DT* before being added to *population*. Similarly, at each time step, *removal* * *DT*, not just *removal*, is subtracted from *population*. Give the formula for *population*(*t*) as listed at the equation level.
- **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 *STELLA*'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.stm*, as *comparative.stm*, and 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, in the menu *Run*, select *Sensi Specs…*. Figure 3.4.3 displays the resulting pop-up menu.

Figure 3.4.3 *Sensi Specs…* pop-up menu



Because we wish to compare graphs for several different growth rates, double-click *growth_rate* under the *Allowable* menu. In the *Selected (Value)* menu, *growth_rate* and its default value (0.1) appear. Change the value of *# of Runs* to 4. Click once on *growth_rate* (0.1) in the *Selected* menu. Enter the *Start* value (0.10) and *End* value (0.13). *STELLA* automatically divides the interval evenly to obtain the values of *growth_rate* for the four simulations. Once satisfied with the list, click *Set*. Figure 3.4.4 displays the resulting menu. While still in the *Sensi Specs…* menu, click *Graph*. By default, the graph type is *Time Series* and *Comparative*. Double click *population* to have *STELLA* plot *population* versus time for each of the four *growth_rate* values. Enter the title "Populations for Rates 0.1 to 0.13", and click *OK*. In the main *Sensi Specs…* menu, click *Table*, select *population* to display, and enter the title "Populations for Rates 0.1 to 0.13". Run the simulation. The resulting graph and the end of the table are as in Figures 5 and 6, respectively. Comparison of the results reveals the dramatic impact on the population of even a 1% increase in the growth rate.

Figure 3.4.4 Values in *Sensi Specs…* pop-up menu

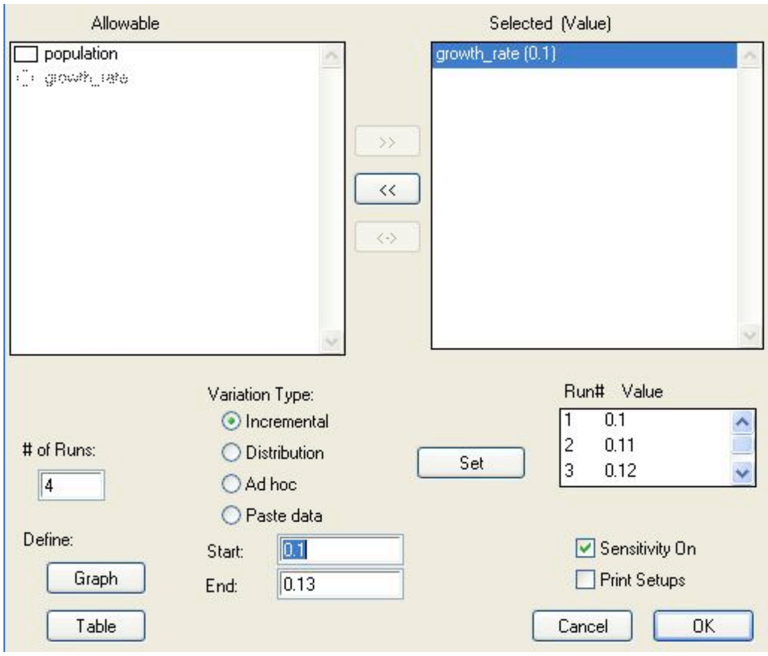


Figure 3.4.5 Graph for comparative simulation

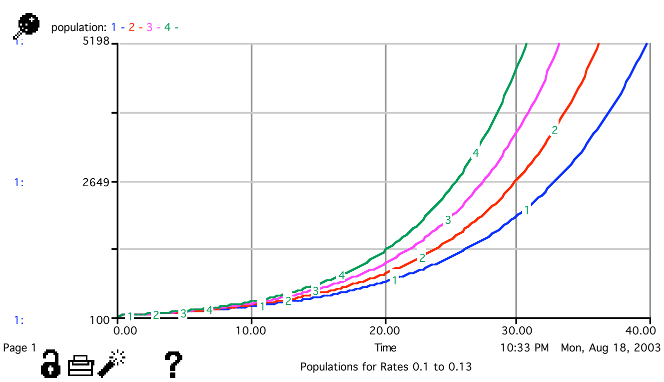


Figure 3.4.6 End of table for comparative simulation

Time	**1: population**	**2: population**	**3: population**	**4: population**
…	…	…	…	…
38.75	4,594.09	6,701.68	9,767.19	14,221.96
39.00	4,708.94	6,885.98	10,060.21	14,684.17
39.25	4,826.66	7,075.34	10,362.02	15,161.41
39.50	4,947.33	7,269.91	10,672.88	15,654.15
39.75	5,071.01	7,469.84	10,993.06	16,162.91
Final	5,197.79	7,675.26	11,322.86	16,688.21

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.stm*, as *graphInput.stm*, and open this new file.

Sometimes we have a concept of the trend of a converter 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. Double-click *growth_rate*; in place of the equation type *TIME*, the independent variable; and click *To Graphical Function* on the lower left of the pop-up menu. We can either enter the raw data in the *Time* and *growth_rate* columns on the right, or we can click on appropriate values on the graph. For example, suppose the growth rate starts at 0.10, decreases to almost 0, and then increases again. Adjust the maximum *growth_rate* graphical value to be 0.1. By clicking on the graph, enter values for growth rate related to time as in Figure 3.4.7. Using the *Edit Input* box, adjust the values as necessary to agree with those in the far right column of Figure 3.4.7. Run the simulation. The resulting graph of *population* versus *time* should appear as in Figure 3.4.8.

Figure 3.4.7 Graphical input

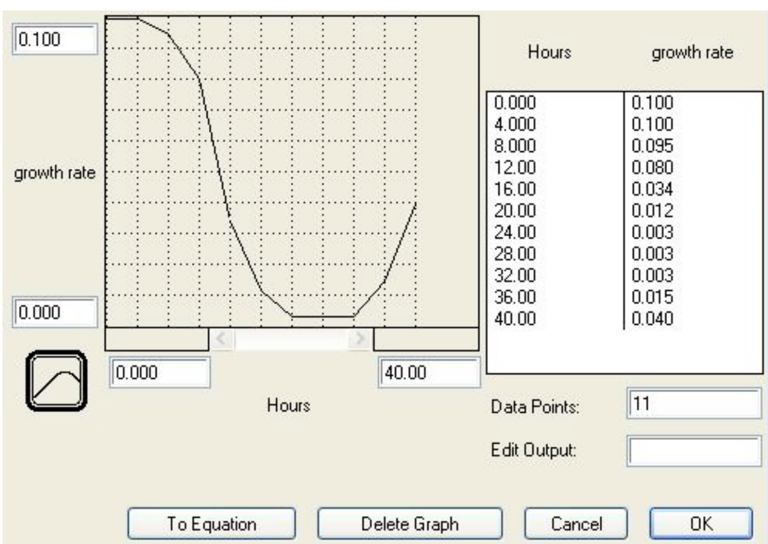
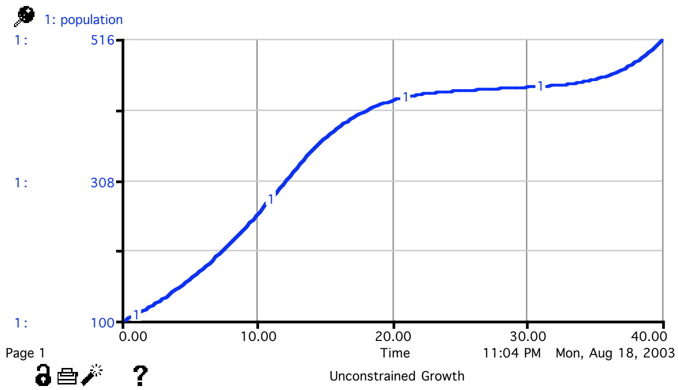


Figure 3.4.8 Resulting graph from *growth_rate* input graph in Figure 3.4.7



Quick Review Question 13 Double-click *growth_rate*. By clicking the graph icon on the lower 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. Describe the shape of the graph and explain the results.

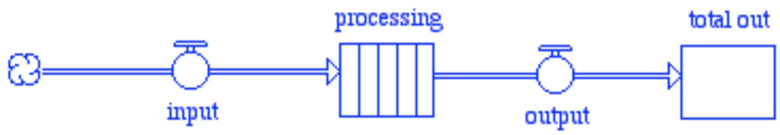
Conveyor

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 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 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. Let the simulation run for the default 12 days with *DT* = 0.25 days. Make the value for *input* be 100 pints per day and the initial value of *total_out* be 0 pints. Double-click *processing*; and using the radio buttons, change the stock type from *Reservoir* to ***Conveyor***. Make the transit time be 7 days and the initial value 0 pints. After clicking *OK*, we see a model diagram similar to Figure 3.4.9 with the conveyor, *processing*, represented with vertical lines through a rectangle. As we see in the next quick review question, we do not need to enter a value or equation for the flow *output*; the conveyor regulates output flow. Save your work in a file called *conveyor.stm*.

Figure 3.4.9 Model with conveyor *processing*



Quick Review Question 14 Double-click *output*. Give the value in the large text box at the bottom of the popup menu and give its meaning.

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

Getting Started with the STELLA *Software, A Hands-On Experience*, a booklet distributed with the software from isee systems that includes a tutorial (http://www.iseesystems.com/)