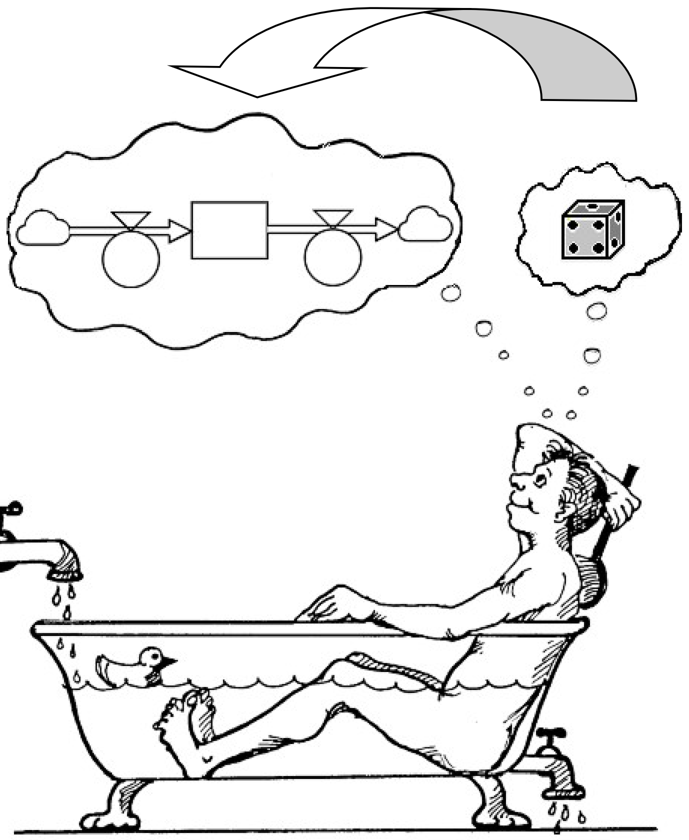
StochSim User's Manual



Part I. Deterministic modelling and simulation with StochSim

Part II. Stochastic modelling and simulation with StochSim

Home Page: https://stochsd.sourceforge.io/homepage/

Contents

Part I. Deterministic modelling with StochSim

1. Introduction

StochSim (Stochastic Simulation) is an open-source tool for deterministic and stochastic modelling and simulation based on Insight Maker[1] version 5. That StochSim is open-source means that you get it and use it for free. It also gives you the right to modify the StochSim source code under open-source licence, see Appendix B. StochSim is written in JavaScript.

StochSim is developed to build, simulate and analyse *deterministic* and *stochastic* modelling and simulation according to the System Dynamics approach. StochSim is also supplemented with tools for sensitivity analysis, optimisation, statistical analysis and parameter estimation.

Part I of this manual focuses on how you build and simulate *deterministic* models in StochSim, while Part II focuses on *stochastic* modelling, simulation and statistical analysis from multiple simulation runs.

StochSim can be used to study time continuous progress in a great number of areas, for example biology, economics, ecology, physics or technology. A model is constructed by placing and connecting predefined graphical symbols at suitable places on the screen. StochSim then transforms the graphical model into an executable code.

StochSim is available in two versions: StochSim Desktop and StochSim Web. Both versions of StochSim work on Windows, Mac OS X and GNU/Linux.

The Desktop version is downloaded from StochSim's homepage and runs as a normal program (using NW.js which stands for Node-Webkit), while the web version requires a web-browser such as Google Chrome or Mozilla Firefox to load StochSim and run your models. StochSim does not support Internet Explorer.

For installation of StochSim Desktop and running of StochSim Web see Appendix A.

Purpose: To build dynamic and stochastic compartment models and simulate their behaviours.

Worldview: The world consists of two fundamental elements: **Stocks** and **Flows**.
cxfd3244
StochSim Home Page: https://stochsd.sourceforge.io/homepage.

[1] Insight Maker is a simulation package developed by Scot Fortmann-Roe. We have taken the open System Dynamics part: simulation engine, function library, error detection facility, macro facility etc. from Insight Maker. However, the non-open parts based on mxGraph (for model flowchart and result diagrams) and Ext JS (for the modelling window and buttons etc.) have been replaced in order to make StochSim completely open. Furthermore, the file handling has been rewritten. Finally, we have made some modifications to conform to the System Dynamics conventions. (Insight Maker is available at: http://InsightMaker.com.)

2. The fundamental building blocks of a StochSim model

Stock is a container for something, e.g. water, people, cars, or money. (Synonyms: **State variable**, **Compartment**, **Level**). A new Stock should be assigned a proper name after double-clicking the symbol.

Flow is the active element that generates changes. A flow is always a *rate* - something *per time unit*. It may be water per second, Immigrants per year, Deaths per month, Cars produced per week, etc. Flows fill up or drain Stocks. (Synonyms: **Transition**, **Rate**). The new Flow should be assigned a proper name after double-clicking the symbol. Note that the arrow shows the *positive direction* of a flow. If the flow rate is negative, it flows at the direction opposite to the arrow.

The **dynamic behaviour** of a model is created by flows accumulated in and/or drained from Stocks! *Flows into Stocks make the world go around!*

Often a ***bathtub analogy*** is used as a metaphor. The Stock is a *bathtub* where e.g. water is added by an Inflow and drained by an Outflow.

A **Stock** can only be changed by *physical flows* (of water, money, rabbits etc.). But also a **flow rate** can change because of *influence* from other elements. This influence (transmitted through an *information link*) is represented by a single arrow and may *not* be confused with a physical flow.

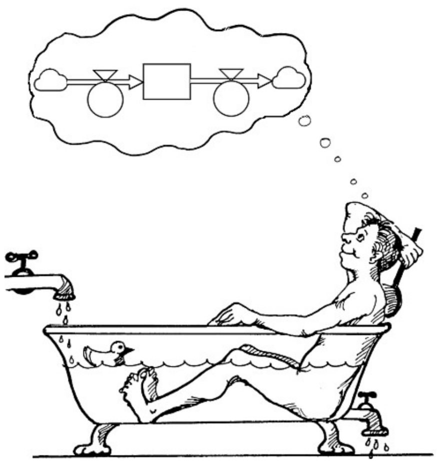


Figure 2.1. The bathtub analogy.

Information link: Influence from one element on another. (Synonymous: **Link**, **Information, Signal**.) A link has no name.

For practical reasons, we add three more graphical symbols.

Variable contains a calculation based on the values of other elements. It can also be a constant that is fixed over time. (Synonym: **Auxiliary**). The variable should be assigned a proper name after double-clicking the symbol.

Constant is a special case of a variable that cannot have any incoming links. It is intended to hold parameters (constants) Of pedagogical reasons, it is good to separate a parameter (who's value is externally defined) from a Variable (that is deduced from other model elements). Constants are named after double-clicking the symbol.

Converter is a table look-up function where you can describe the relation between two elements like X and Y in tabular form. See 11.2.6.

3. Starting up StochSim

StochSim is available in two versions: **StochSim Desktop** and **StochSim Web**. For installation and use, see Appendix A. This manual applies for both versions.

When you open StochSim, the StochSim screen is then shown. On the upper part of the screen you see the main menus and some buttons, and below is the model window (which is empty at start), where you build your model in a click-and draw manner.

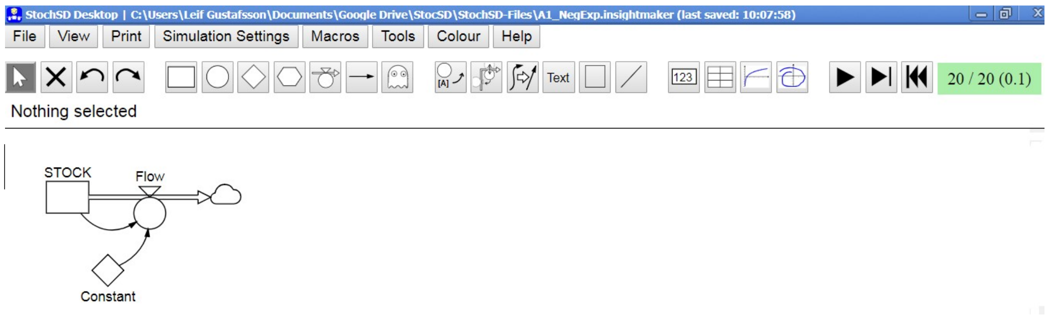

StochSD Desktop | C:\Users\Leif Gustafsson\Documents\Google Drive\StocSD\StochSD-Files\A1_NegExp.insightmaker (last saved: 10:07:58)
File
View
Print
Simulation Settings
Macros
Tools
Colour
Help
Text
123
20 / 20 (0.1)
Nothing selected
STOCK
Flow
Constant

Figure 3.1. The main menu of StochSim located above the modelling window (here shown with a few building blocks).

The main menu of StochSim consists of pull-down menus and a row of buttons for constructing and handling the model.

3.1 MENUS

In the pull-down menus you find: **File**, **View**, **Print, Simulation Settings**, **Macros**, **Tools**, **Colour** and **Help**.

The File menu

The file handling in StochSim is handled from the File menu in a similar way to most programs, e.g. Microsoft Word.

In StochSim, a model constitutes a file. You can *create* a new model, *save* the model and *open* it again.

When run locally, you can choose the folder to save the model in.
When StochSim runs on a server the files are always stored in the *Downloads* folder.

The File menu contains the usual facilities:

- New
- Open
- Save
- Save As …
- Recent Files

To create a new model you click the **New option**button in the **File** menu.

A model can be opened by clicking the **Open** option in the **File** menu (or directly: Ctrl-O).

By pressing the **Save as …** option in the **File** menu, you open a form for saving your model. Her you can give the file a proper name. In the Desktop version of StochSim you can also browse to a dictionary of your choice, while in the Web version the model is always stored in the *Download* library.

When the model has a name and location, you can use **Save** option in the File menu (or directly use; Ctrl-S).

Up to five **Recent Files** are directly accessible. (Only for StochSim Desktop version.)

The View menu

With the View menu you can customise the size of StochSim on your screen.

- Zoom in (Ctrl +)
- Zoom out (Ctrl -)
- Reset zoom (Ctrl 0)

The Print menu

Printing the model diagram or its underlying equations is performed from the Print menu. This menu contains:

- Print Diagram (Ctrl-P)
- Print Equations

The model diagram can be printed on paper with the **Print** button or Ctrl-P.
The equations can be listed and then printed on paper.

Simulation Settings menu

Before you execute the model, you should specify what time period to study and the size of the TimeStep (Often denoted DT). This is done in the **Simulation Settings menu**, where you specify the *Simulation start*, *Simulation length*, and *TimeStep* between recalculations of the model state.

- Start Time
- Length
- Time Step

Macros menu

Enables the introduction of macros to StochSim, see Section 11.

Tools menu
In the Tools menu, you find tools for *Optimisation*, *Sensitivity analysis, Statistical calculations and presentation* and *Parameter estimation.* Pressing a tool option will make the tool appear in a window on the right-hand side of the screen. With the Hide option you can close the tool.

The Tools menu has the following options:
- Optim
- Sensi
- StatRes
- ParmVar
- Hide

The tools will be treated in Part II.

Colour menu
After marking one or several elements, you can open the Colour menu and select one out of 12 colours.

Note that the colour in a diagram will get the colour of the element.

Help menu
The Help menu contains:
- About StochSim
- User's manual
- Debug log
- Restart and clear model
- Restart and keep model

'About StochSim' briefly describes the current version of StochSim. The 'User's manual' you are now reading is also available here. The 'Debug log' is only valuable for a developer of the software. Finally, if StochSim is malfunctioning, you can restart it with or without the current model loaded.

3.2. BUTTONS

The buttons are from left to right:

Manipulation buttons
These buttons are for recapturing the 'unloaded mouse', deleting a block, or undoing or redoing an operation.
- Mouse
- Delete
- Undo

- Redo

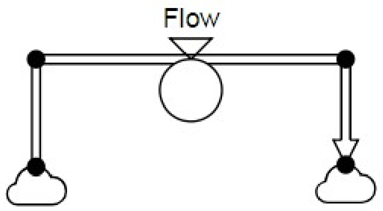
Up to 10 undo steps are remembered and con be redone with the **Undo** and **Redo** buttons.

Building blocks (Elements)

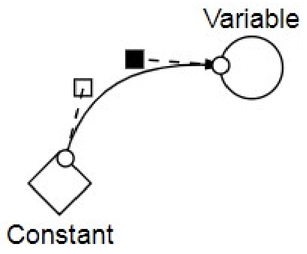
Here are the building blocks of a StochSim model.

- Stock
- Variable
- Constant
- Converter (for a table look-up function)
- Flow
- Link
- Ghost

The **Flow** can be broken by clicking the *Right Mouse Button* when drawing it. Then anchor points will show up when the flow is marked. With these anchor points you can adjust the flow. You may also move the name around and flip or move the valve symbol as described in “Style blocks” below.



A **Link** can be bent into a nice Beziér curve. When you click on a link two anchor points and two handles will be visible. Then you can grip the flow and bend it. Alternatively, you can grip one of the handles (the marked (black) in the figure) and adjust the link.

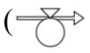
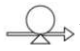


Style blocks

The style blocks provides blocks for styling the model by moving the names of the building blocks, moving the valve symbol of the flow and for inserting frames, texts and lines.

- Move label around the element
- Move valve symbol
- Straighten a link
- Text
- Frame
- Line

The label can be moved to a position North, West, South or East of its element.

The valve symbol can be flipped ( or ).

In a *broken* flow, the valve symbol can be placed at the different segments.

An ugly link can be straightened by marking it and pressing the Straightening button.

Result blocks
These blocks provide Number boxes, Tables, Time Series and xy-Plots, for presenting results.
- Number box
- Table
- Time diagram
- xy-Plot (Plots Y versus X)

Execution buttons
The execution buttons controls the simulation.
- Run/Halt
- Step
- Stop

Time display

To the right of the Execution buttons you find a display showing:
Current Time / End time (Time Step).

4. Model building

4.1 Placing the building at the model window

The main elements for building a model are: *Stock, Variable*, *Constant*, *Converter*, *Flow*, *Link* and *Ghost*.

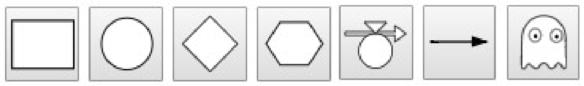


Figure 4.1. The buttons for StochSim building blocks: *Stock, Variable*, *Converter, Flow*, *Link* and *Ghost*.

The six first mentioned elements are described in Chapter 2. With the Ghost button you can duplicate any of the first five elements to be displayed twice.

The building blocks are attached to the Model window and connected to each other in a click-and-draw manner.

It is usually practical to start by placing and naming the *Stocks* – which are the elements you usually (but not always) are most interested in. Then draw and name the *Flows* into, out from or between Stocks. Thereafter, place and name *Variables*, *Constants* and *Connectors*. Finally, place the Links to connect elements.

To place a *Stock*, *Variable*, *Constant* or *Converter*, click on its button, place the mouse cursor at an appropriate place in the Modelling window and drop it by clicking again.

To create a *Flow*, click on the Flow button, place the mouse cursor, press the *left* mouse button to anchor the starting point, hold the button down and draw the mouse in horizontal or vertical direction until it reaches its destination where you release the left mouse button.

A **Flow** has a start and an end point that may connect to a Stock or it will start and/or end with a cloud symbol to show that the flow content comes from or goes to somewhere outside the model boarders (i.e. the models environment). If the start or end point is over a Stock, it will be connected to this Stock and the cloud disappears.

A *Flow* can also be *broken* in x and y directions. This is obtained by clicking the right mouse button during the drawing operation. The flow may afterwards be adjusted by clicking the *Flow* and using the handles (at star, end and at points where the Flow is broken. See Section 3.2.

A **Link** is used to connect two elements. It means that one element affects the other. The *Link* is obtained by clicking its button. Then place the mouse cursor over a *Stock*, *Variable*, *Constant* or *Converter* and draw it to another of these building blocks – but not to the *Constant* (that cannot take incoming links). The link can be shaped using Beziér curves as described in Section 3.2.

To create a *Ghost* of a *Stock*, *Variable*, *Constant* or *Converter*, first click on the actual building block in the Model window, then click on the Ghost button and finally click on an appropriate place in the Model window. The Ghost is then graphically represented as a copy of the original element with a ghost symbol inserted.

The *Ghost* means that the same *Stock*, *Variable*, *Constant* or *Converter* may be represented at two or more places in the model. A *Ghost* cannot have any flows or incoming links - but it can have outgoing links! This is to prevent the logics of the model be spread out.

Using Ghosts can eliminate connecting links to cross over the model that makes the model look ugly and unstructured. You may also use Ghosts to assemble information from Constants (parameter values) or Stocks from different parts of a model into an ‘instrument panel of Ghosts and Number boxes (see Chapter 8).

The elements on the screen can afterwards be adjusted by drawing them to new positions.

The ***names*** of the symbols: *Stock*, *Variable*, *Constant*, *Converter* and *Flow* are always starting with ‘[‘ and ending with ‘]’. For example [Stock1], [Flow2], [Constant1]. The default names should be renamed to be descriptive, e.g. [Rabbits], [Births per month], [Fertility]. A Link has no label and the Ghost gets the same name as the symbol it ghosts. All names can be rotated around its symbol with the ‘Rotate name’ button (the first of the Style buttons), see Section 4.2.

4.2 Adding Styling blocks

Finally, there are a number of *Style buttons* for replacing the name of a building block, for moving the valve symbol of a flow and for including texts, frames and lines into the model.

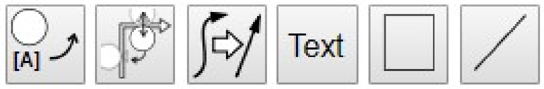


Figure 4.2. The Style elements: Move label, Move valve, Straighten link, Text, Frame and Line.

The name of a *marked* element can be relocated to North, West, South and East of the symbol.

The ‘valve part’ of a *marked* flow can be flipped upside-down and in a ‘broken’ flow be located to different segments of the flow.

The third button Straightens a marked link. This can sometimes be more convenient than adjusting Anchor points or Handles.

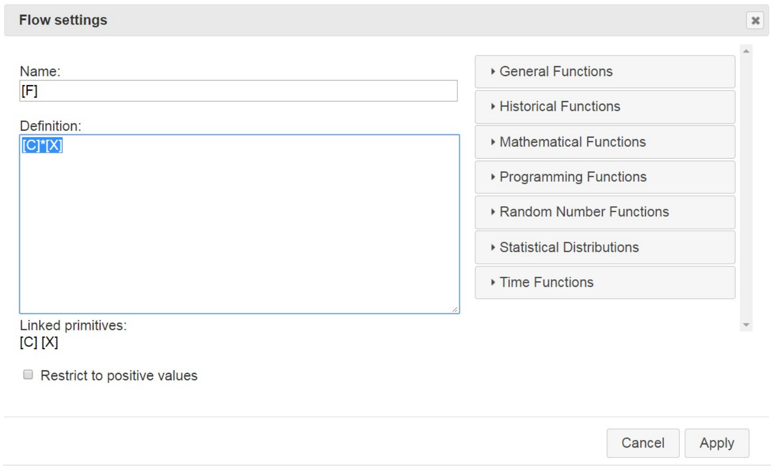
Text frames, Empty frames and straight Lines are created by clicking respective button and drawing its extension at a proper place in the Model window. The size of these blocks can also be adjusted afterwards.

4.3 Naming and defining relations or values of the elements

In StochSim a ***name of an element*** *always includes brackets, e.g.* [Stock1], [WaterFlow], [c]. *However, in the model window and in tables and diagrams, the brackets are not displayed of aesthetic reasons.*

When an element is placed, it is time to give it an appropriate *name* and to define its value or relation to other elements in a *formula*.

Double-clicking an element opens its **Dialog box** (form) where you can specify the elements *name* and define its *formula*. Figure 4.3 shows the Dialog box for a Flow.


Flow settings

Name:
[F]

Definition:
[C]*[X]

Linked primitives:
[C] [X]

Restrict to positive values

Figure 4.3. Dialog box of a Flow with: 1) Name field where the name can be changed, 2) Value field where the value or function is specified. 3) Linked elements show the elements that should be included in the Value field. 4) library functions which also could be used in the Value field. 5) Check box for Restriction to positive values. 6) Cancel or Apply buttons.

For the *Stock*, *Variable*, *Constant* and *Flow* elements the Dialog boxes are similar. (A *Ghost* has *the same* (not just a copy) dialog box as the element it ghosts.)

The **Dialog box** contains a **Name** field where the name (e.g. [Stock3], or [Variable5]) can be changed to a suitable name in upper or lower case characters. If you skip the brackets ('[' and ']') StochSim will automatically insert them.

Below is the **Definition** field of the element where you define a formula or a value. Here, linked elements and library functions can be used together with numbers and the arithmetic operators + - * / ^ and ( ).

Further below, **Linked primitives (elements)** to be used in the Definition are listed. They enter the Definition field when you click them.

For Stock and Flow elements a check box '**Restrict to positive values**' is found at the bottom-left. (This should *very seldom* be used).

To the right you find, **Library of functions** subdivided into groups named: *General Functions, Historical Functions*, *Mathematical Functions*, *Programming Functions*, *Random Number Functions*, *Statistical Distributions* and *Time Functions*. Clicking on such a group button will give you a list of functions in the category. These functions will be treated in Section 10.

StochSim knows how to calculate the value of a **Stock** from inflows to and outflows from a Stock from the graphical structure without further specification. Therefore, a Stock only requires the initial value at '*Time zero*' (i.e. the time when the simulation starts) in the definition field. Usually, the initial value for a stock is a number. But it is also possible to initiate the Stock to the value of a Variable or Constant linked to the Stock. The initial value may also be a random number.

The **Converter** is a table look-up function where you specify the pairs: x_1,y_1; x_2,y_2, …, x_n,y_n of values. It has its own dialog box. See Section 10.2.6.

A **Link** has no dialog box. It just links one element to another.

Leave the dialog box by clicking the **Apply** button to realise your specifications or by clicking the **Cancel** button to leave the dialog box without new specifications.

Note, if a formula (say [c]*[X]) that is linked to another element (say [X]) and you then change the name of that element (from [X] to [Y]), then the list of linked variables is updated – but not the formula in the Definition field is not. (The formula still refers to the non-existing name [X]). You must then also update the formula. See Figure 4.2.

5. Equations

When you defined all relations and values related to the elements, StochSim automatically generates the equations that mathematically describe the model.

The model equations can be viewed by opening the **Print menu** and choose **Print Equations**. You will then see the **Equation List**. (From here you may also print the equations.)

6. Specification of the simulation

Before you execute the model, you must *specify* what *time period* to study and *how frequently recalculations of the model should be performed*.

The specification of the time handling is done in the **Simulation Settings menu**. Here you specify *Start*, *Length*, and *TimeStep* for the simulation.

The *integration* (updating of the model over time) is performed by Euler's algorithm. This algorithm updates a Stock X over time according to: X(next) = X(now) + Inflows(during now to next) – Outflows(during now to next). Mathematically this is expressed as:

$$X(T+DT) = X(T) + DT*InFlowRate – DT*OutFlowRate.$$

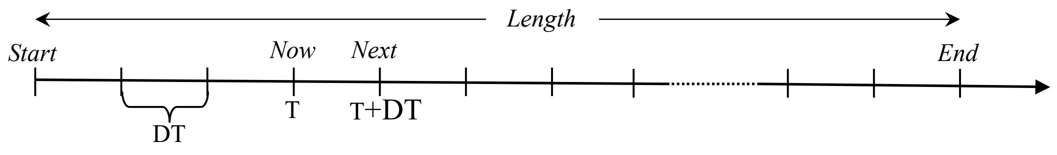


Figure 6.1. The time-handling relating *Start*, *Length* and *TimeStep* (DT).

For a 'well-behaved' deterministic model there is more efficient algorithms than Euler's, but for stochastic models the gain from these algorithms are small or none.

Note that a short time-step increases the accuracy of the calculations but also increases the number of recalculations of the model and thus also the execution time. You should, therefore, try to find a compromise where the time-step (DT) is sufficiently short to give good accuracy - but not shorter than this.

7. Simulation

A simulation of the model is performed by pressing the **Run/Halt** button. It is also possible to test the model behaviour with the **Step** button. You may also reset the calculations with the **Reset** button.

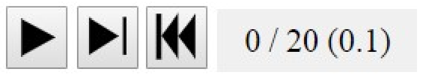


Figure 7.1. The execution buttons: **Run/Halt, Step and Reset**. (To the right of the buttons the **Actual simulation time** or **Start Time / End Time** and **(Time Step)** are displayed.

Pressing the **Run** button (▶) starts the simulation and enables specified *Simulation Results* in Number boxes, Tables, Time diagrams or xy-plots. See Chapter 8.

The calculations behind the scene of a simulation

What happens during a simulation is that StochSim, time-step by time-step, solves the equation system. The working procedure used is the following (DT is the simulation time-step used and EndTime is the StartTime + Simulation Length):

```
1. Time = StartTime
2. The Stocks are initiated
3. GoTo 5

4. The new Stock values are calculated
5. The new Variable values are calculated
6. The new Flow values are calculated
7. The results for this time-step are stored
8. Time = Time+DT
9. If Time < EndTime then GoTo 4
```

Figure 7.2. The calculations performed behind the scene. The calculations of the Stocks are updated according to Euler's integration method.

8. Results presentation

To see the results of a simulation run you can use **Number boxes**, **Tables**, **Time diagrams** and **xy-plots.**

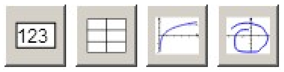


Figure 8.1. Result presentation buttons for **Number box**, **Table**, **Time diagram** and **xy-plot**.

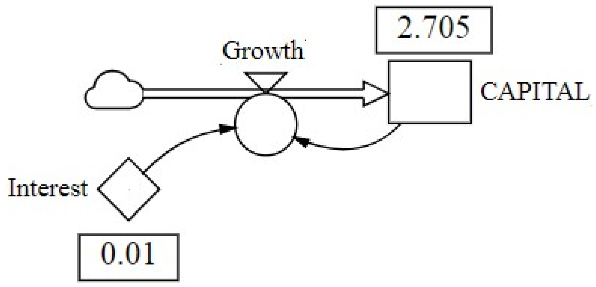
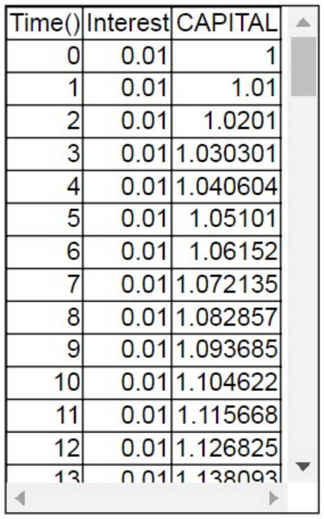


Figure 8.2. The Number box is particularly useful to show the values of parameters (Constants) and the end results of Stocks. Here capital growth of 1 Euro at 1 % interest after 100 years is shown. (DT=1 to create annual increase.)

Tables, **Time diagram** or **xy-plot** is obtained by clicking respective result button. Then you go to a proper place on the screen and click-and-draw the item to a proper size. Tables and diagrams can also afterwards be drawn to a proper size by using the handles.

By double-clicking the **Table**, **Time diagram** or xy-plot, you can choose the values of the elements to be displayed over the simulation run. You can also choose to only display what happens during a selected period of the simulation run and how often the results should be tabulated or plotted. The Time diagram and the xy-plot are automatically scaled after a simulation.



Time()	Interest	CAPITAL
0	0.01	1
1	0.01	1.01
2	0.01	1.0201
3	0.01	1.030301
4	0.01	1.040604
5	0.01	1.05101
6	0.01	1.06152
7	0.01	1.072135
8	0.01	1.082857
9	0.01	1.093685
10	0.01	1.104622
11	0.01	1.115668
12	0.01	1.126825
13	0.01	1.138093

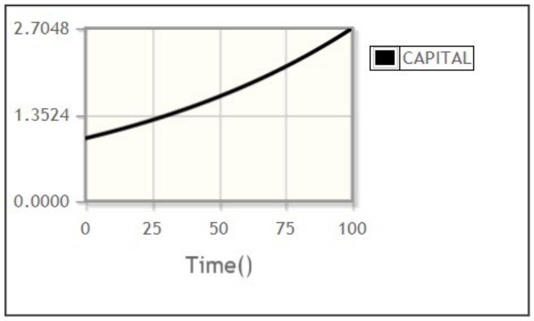


Figure 8.3. Table and Time diagram.

Note that you select the colour of the line or mark in a **Time diagram** by colouring that element. Having the same colour of the element and its line in a plot reduces the risk of misunderstanding. In a Time diagram, a Stock will be drawn with a *solid line*, a Flows with a *dashed line* and an other type of element with a *dotted line*.

For Tables you can also choose to print only a part of the data and the time between the printings.

A **xy-plot** shows one quantity (y-axis) varying against another quantity other than Time (on the x-axis).

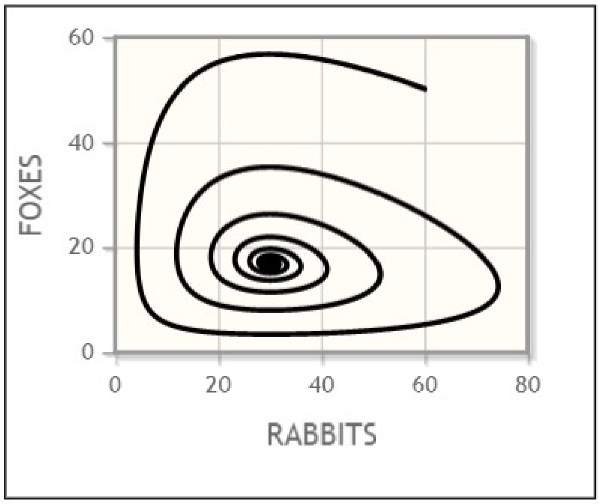


Figure 8.4. A xy-plot of Rabbis versus Foxes in a prey-predator model.

In a **xy-plot** the you can choose between *Line* and *Mark*. The colour is black.

9. Examples

Example 9.1: Filling a bathtub with an open outlet.

Construct the following model to see what happens when water flows into and out from a bathtub.

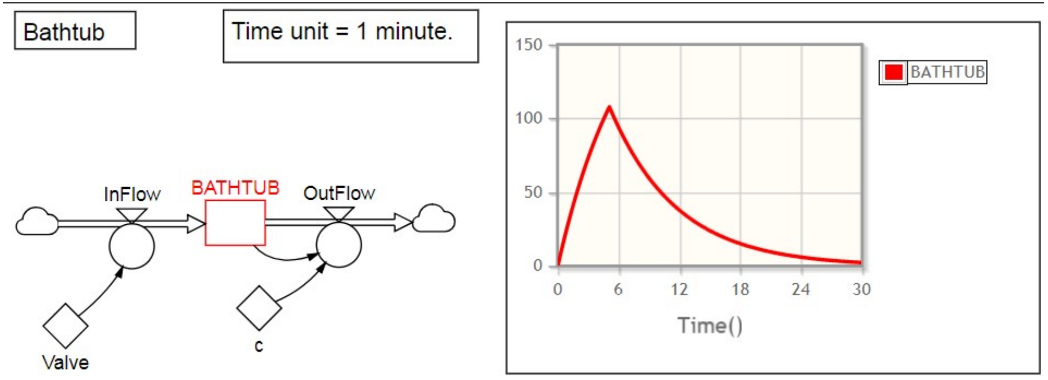


Figure 9.1. A simple bathtub model and its behaviour.

We here assume the following:
1) The BATHTUB is empty at Start Time.
2) The InFlow is regulated from the Valve.
3) The Valve handle is opened to give 30 litres/minute for *5* minutes and then closes.
4) The OutFlow is open. For simplicities sake, we assume that 15% of the amount of water in the BATHTUB will leave the tub each minute through the OutFlow.

The equations generated from the model are:
[Valve] = 30
[c] = 0.15
[BATHTUB] = [BATHTUB]+DT()*([InFlow] – [OutFlow])
[BATHTUB] = 0 (* Initial value *)
[InFlow] = IfThenElse(T()<5, [Valve], 0) (* See Section 10.2.1 *)
[OutFlow] = [c]*[BATHTUB]

The model is repeatedly updated time-step by time-step until the simulation is complete.

Using *the time unit = 1 minute*, the time-step (DT) is set to *0.1* minute, so the equation system becomes updated for each *6* seconds.

You can choose the **time unit** to be e.g. *1 second*, *1 minute* or *1 hour*. But you have to be *consistent* and *recalculate* all time data to the selected one!

The content of the bathtub is shown for half an hour (30 minutes) in a Time Diagram in Figure 9.1 above. ■

Example 9.2: Rabbits on an isolated island.
We wish to study the progress of a rabbit population on an isolated island in the sea where *10* rabbits are introduced. They eat, breed and die of old age. Studies of the system have shown that the food supply available is constant and is estimated to *100* kg of carrots per month. It has also been shown that the number of rabbits born per month is proportional to the size of the population and to the *square root* of the amount of food per rabbit and month. The fertility constant has been calculated to *0.2*. Further, it has been found that the average length of life is *20* months, which is about the same as *5%* of the population dying each month.

Figure 9.2. Picture of the system under study when dropping the carrots.

A so-called causal-loop diagram for the model is shown in the figure below.

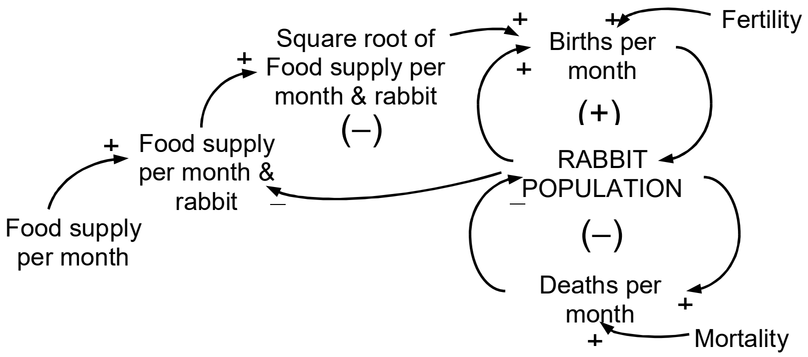


Figure 9.3. A causal-loop diagram of the rabbit system.

Building the model structure and filling in all relations and parameters (the fertility and mortality constants are f = *0.2* and m = *0.05*, respectively) gives the following StochSim model:

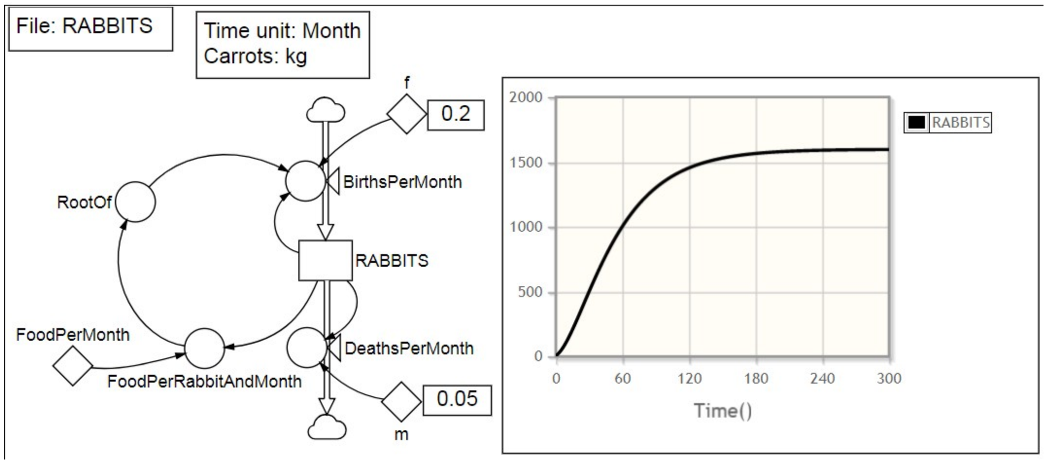


Figure 9.4. A StochSim diagram of the rabbit system and the result.

Select the **Simulation Settings** menu and set the *Simulation Length* to *300*, chose *Months* as *Time unit* and set the *Simulation Time-Step* to *0.5* (months).

When the model is complete, and the **Run** button is pressed, we choose the **Time Diagram,** here only showing RABBITS, see Figure 9.4, above. We could also choose **Table** and e.g. select *RABBITS*, *BirthsPerMonths*, *DeathsPerMonths*, and *FoodPerRabitAndMonth*. ■

10. Functions

In StochSim there are a large number of library functions. You find them to the left of the Dialog box of a **Stock**, **Flow**, **Variable** or **Constant**. The functions are first listed as an overview. A more detailed presentation is given below.

The arithmetic operators are: + – ∗ / ^ ( ) for addition, subtraction, multiplication, division, power (x^n) and parentheses. As always, the calculation order is: power, then comes multiplication and division, and last addition and subtraction. Parentheses '( )' can be used to alter this order.

10.1 The categories of functions in StochSim

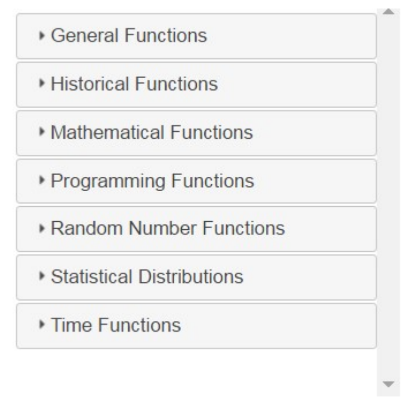


Figure 10.1. The categories of functions.

Below, we will shortly present the most commonly used functions sorted into different categories.

Only the most useful functions will be discussed. Sorted by category, these are:

General Functions
- If Then Else
- PulseFcn
- Step
- Ramp
- Stop (Terminates the simulation run)

Historical Functions
- Delay1 (Dynamic delay of order 1)
- Delay3 (Dynamic delay of order 3)
- PastMax
- PastMin
- PastMean
- PastStdev
- PastCorrelation
- Fix (A device to sample and hold values for regular periods of time)

Mathematical Functions
- Abs (Absolute number)
- Sqrt (Square root)
- Sin (Argument in radians)
- Cos (Argument in radians)
- Tan (Argument in radians)
- Log (Logarithm base 10)
- Ln (Logarithm base e)
- Exp (Exponential)

Programming Functions
- If-Then-Else (More advanced version than that in *General Functions*.)

Random Number Functions

• PoFlow	(Simplified expression for Poisson flows)
• Rand	(Uniform distribution)
• RandNormal	
• RandBinomial	
• RandPoisson	
• RandExp	
• RandBoolean	
• RandDist	(Custom distribution)

Statistical distributions
In 'Statistical Distributions' you find Pdfs, Cdfs and Inversions of *Normal*, *Lognormal*, *t*, *F*, *Chi square*, *Exponential* and *Poisson* distributions. These distributions can be used in some advanced statistical studies. However, these distributions will not be further discussed in this manual.

Time Functions
- Current Time
- Time Start
- Time Step
- Time Length
- Time End

Converter
- The often-useful empirical graph function (sometimes called table-look-up function) where you can enter an x-value to obtain the y-value is named *Converter*, and is an element with its own button. However, since it is a function it is discussed (last) in this chapter.

10.2 Detailed description of the functions

10.2.1 General Functions

IfThenElse – Simple 'If Then Else' function

Syntax:	IfThenElse(Test Condition, Value if True, Value if False)
Result:	If the logical *Condition* is True then the first *Expression* (*Value*) is evaluated and returned, if it is False then the second *Expression* is evaluated and returned.
Example:	If(A < B, A, B) returns the smallest of A and B.
Example:	If(A < B, 5, 2*Sqrt(25)) returns 5 if A < B, otherwise 10.

Logical operators are: <, >, <=, >=, = and <> (‘not equal to’), AND, OR.

Syntax: A < B, A > B, A <= B, A >= B, A = B, A <> B, X>=A AND X>B

Example: A <> B will return True if $A \neq B$ and False if A = B.

PulseFcn – Pulse function

Syntax: PulseFcn(Start, Volume, Repeat)

Result: Creates a pulse input at the specified *Time* with the specified *Volume*. *Repeat* is optional and will create a pulse train with the specified *Repeat* as interval if positive. With a negative *Repeat* value you get only a single pulse at *Start*.

Example: Pulse(2, 1, 4) generates repeated pulses with the *Volume*: 1, starting at time=2 and then repeating each 4 time units, see Figure.

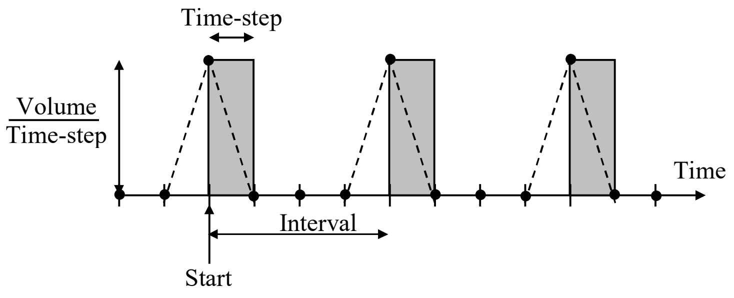


Figure 10.2. The PulseFcn.

An ideal pulse is infinitely high and infinitesimally wide, so that the Height*width equals the specified Volume. The smaller time-step - the better the approximation of an ideal pulse.

The graph in the figure is shown for the time-step = *1* time unit. (For e.g. time-step = *0.1* the pulse will be 10 times higher and have a tenth of the width. The interval between the pulses will still be *4 time units – which now is 40* time-steps.)

Note the difference between what happens mathematically (grey rectangles) and what it looks like in a graph where the lines (dashed) are connected to the values (dots) at each time-step.

[The PulseFcn replaces the awkward Pulse in Insight Maker where the pulse-content (volume) changes when you adjusted the time-step.]

Step - Step function	
Syntax:	Step(Start, Height)
Result:	Generates a step at time *Start* with amplitude *Height*. That is, it returns 0 if TIME < Start, and *Height* otherwise.

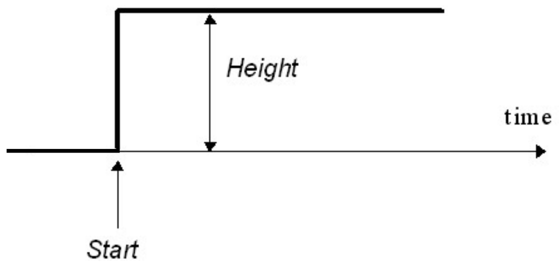


Figure 10.3. The Step function.

Ramp - Ramp function	
Syntax:	Ramp(Start, Finish, Height)
Result:	Generates a ramp, which moves linearly from *0* to *Height* between the *Start* and *Finish* times. Before *Start* the value is *0*; after *Finish* the value is *Height*.

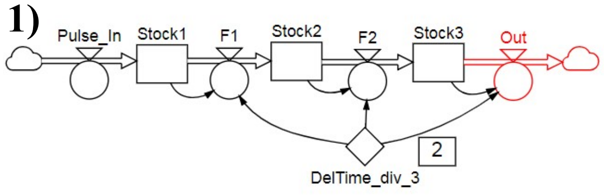
Stop – Stop function	
Syntax:	Stop()
Result:	Immediately terminates the simulation run when executed. The Stop function can be used in an IfThenElse function.
Example:	IfThenElse([InfectiousPersons] < 0.5, Stop(), 0) [When the number of infectious persons goes below a certain value (e.g. 0.5) the simulation run terminates.]

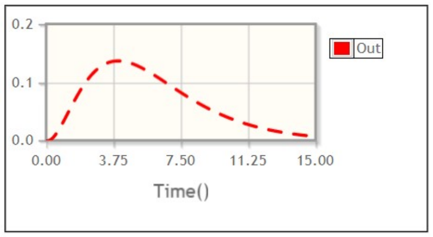
10.2.2 Historical functions

<u>*Short Introduction to delays*</u>
A delay is, for example, the time from which a product is ordered until it is delivered, or the time from which a person becomes infected until he becomes sick. Also information can be delayed, for example, the time to deliver a letter.

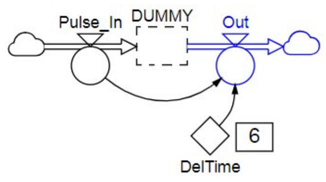
A delay is a dynamic subsystem that contains a series of stocks and flows. The number of states in that subsystem is the *order* of the delay. See the Figure 10.4.

1)





2)



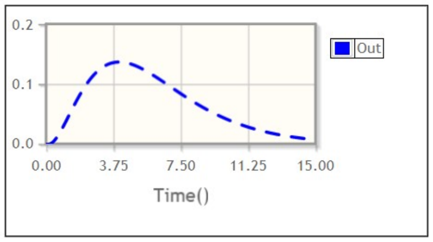


Figure 10.4. A delay of order 3 (with e.g. a total delay time of 6 time units) can be obtained in two ways: **1)** By connecting three stocks in a series, where each of the three stocks is delayed for Del/3 time units.
2) By using the *Delay3* function for the outflow '*Out*', so [O*ut*]= Delay3([Pulse_In], [Del], 0). This connects *Pulse_In* to *Out*. (A stock, here called *DUMMY* because it is not involved in the delaying, may or may not be included. It is merely placed between the inflow and the outflow, so that it is filled by *Pulse In* and drained by *Out*. *DUMMY* will then have the same content as *Stock1+Stock2+Stock3*.)

The order of the delay determines how quickly the output *starts to change* after a change in the input. Low-order delays *start to* respond more quickly than high-order delays having the same average delay time Del. See Figure 10.4, above.

Delay1 – Delay function of order 1

Syntax: Delay1([Element], Delay Length, Initial Value)

Example: Delay1([Inflow], 6, 0)

Delay3 – Delay function of order 3

Syntax: Delay3([Element], Delay Length, Initial Value)

Example: Delay3([Inflow], 6, 0)

PastMax – Maximal value over a past period. For statistics over (a part of) a simulation

Syntax: PastMax([Element], Period = All Time)

Results: Calculates the maximal value of an element over the specified past *Period*. If the *Period* is omitted, it shows the maximum sofar over the whole replication.

Example: PastMax([X]) will create a function that displays the first value of X after one time-step, the largest value of X in the past two time-steps, and so on until the replication is over when the largest value of X is shown.

Example: PastMax([X], [5 Days]) will create a function that starts as in the example above, but after 5 days and more displays the largest value during the last five days.

PastMin – Minimal value over a past period. For statistics over (a part of) a simulation

---

Syntax: PastMin([Element], Period = All Time)

Results: Calculates the minimal value of an element over the specified *Period*.

PastMean – Mean value over a past period. For statistics over (a part of) a simulation

---

Syntax: PastMean([Element], Period = All Time)

Results: Calculates the mean value of an element over the specified *Period*.

PastStdev – Standar deviation over a past period. For statistics over (a part of) a simulation

---

Syntax: PastStdev([Element], Period = All Time)

Results: Calculates the standard deviation of an element over the specified *Period*.

PastCorrelation – Correlation between two quantities over a past period. For statistics over (a part of) a simulation

---

Syntax: PastCorrelation([Element], Period = All Time)

Results: Calculates the correlation between two quantities over the specified *Period*.

Fix – A device to sample and hold values for regular periods of time.

Syntax: Fix(Value, Period)

Results: A device to sample and hold values for regular periods of time.

Example: Fix(Rand(0,10), 5) will draw a random number that remains fixed for the following 5 time units.

10.2.3 Mathematical functions

Syntax	Function	Comment
Round(X)		Rounds a number to its nearest integer
Abs(X)	abs(X)	The absolute value of X
Sqrt(X)	$\sqrt{X}$	
Sin(X)	sin X	Note: X in radians
Log(X)	$\log_{10} X$	The 10-logarithm
Ln(X)	ln X	The e-logarithm
Exp(X)	e^x	
(X) mod (Y)	X (mod Y)	Returns the remainder of X/Y **Example:** 13 (mod 5) = 3

10.2.4 Random Number Functions

Here the most common random number functions are presented. In many simulation languages, random number functions include a *seed* that defines the initial value of the random number sequence. In StochSim the seed is set globally, which can be done by a macro. See example in **Chapter 11. Macros**.

PoFlow - Poisson distributed flow
(This function is an addition to the Insight Maker functions.)

Syntax:	PoFlow(FlowRate)
Result:	Generates Poisson distributed random flow with a mean of *FlowRate*. The result is always an integer ≥ 0.
Example:	PoFlow([c]*[X]) is a shorter, more practical and mor readable form for RandPoisson(DT()*([c]*[X])/DT().

PoFlow() can be used to produce a random number of events during a time-step in a flow.

Rand - Uniformly distributed random numbers	
Syntax:	Rand (Min, Max)
Result:	Generates uniformly distributed random numbers between *Min* and *Max*. Rand() gives uniformly distributed random numbers between *0* and *1*.
Example:	Rand(5, 17) gives uniformly distributed random numbers between *5* and *17*.

RandNormal - Normally distributed random numbers	
Syntax:	RandNormal(Mean, Standard Deviation)
Result:	Generates normally distributed random numbers with a specified *Mean* and *Standard Deviation*.
Example:	RandNormal(3, 0.5) gives normally distributed random numbers with a mean of *3* and a standard deviation of *0.5*.

RandBinomial - Biomially distributed random numbers	
Syntax:	RandBinomial(Count, Probability)
Result:	Generates binomially distributed random numbers from *Count* trials, each with the specified *Probability*. The result is always an integer ≥ *0*.
Example:	RandBinomial(10, 0.3) gives binomially distributed random numbers between *0* and *10* with an expected value of $10 \cdot 0.3=3$.

RandPoisson - Poisson distributed random numbers	
Syntax:	RandPoisson(Lambda)
Result:	Generates Poisson distributed random numbers with a mean of *Lambda*. The result is always an integer ≥ 0.
Example:	RandPoisson(5) gives Poisson distributed random numbers with a mean of *5*.

RandPoisson can be used to obtain the number of events during a time interval.

RandExp - Exponentially distributed random numbers

Syntax:	RandExp(Beta)
Result:	Generates exponentially distributed random numbers with a rate *Beta*. The result is always an integer ≥ 0.
Example:	RandExp(10) gives exponentially distributed random numbers between *0* and *infinity* with a mean of *Beta=10*.

RandExp can be used to obtain the distance in time (or space) between successive events when the rate of events is Beta.

RandBoolean - Binary distributed random numbers

Syntax:	RandBoolean(Probability)
Result:	Generates exponentially distributed random numbers that returns *1* with the specified *Probability*, and *0* otherwise.
Example:	RandBoolean(0.5) can be used for the flipping of a symmetric coin.

RandDist - Custom distributed random numbers

Syntax:	RandDist(x, y)
Result:	Generates random numbers according to a custom distribution. The custom distribution is specified by a sequence of (x,y)-coordinates. The Points between are linearly interpolated. The distribution does not have to be normalised such that the area under the curve is *1*, but the points must be sorted from the smallest to the largest x.
Example:	RandDist({0, 1.2}, {2, 2.3}, {3, 5.3}, {4, 3.3}).

10.2.5 Time functions

T – Current time

Syntax:	T()
Result:	Gives the current time for the simulation.

TS – Start time for the simulation

Syntax:	TS()
Result:	Gives the start time for the simulation. This value is specified in the **Simulation Settings** form.

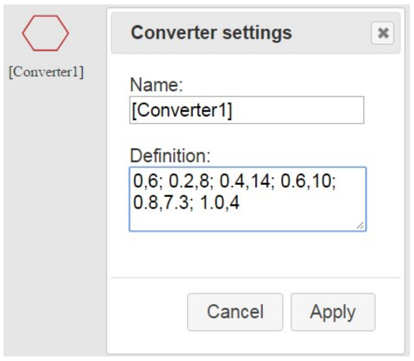
DT – Step-size	
Syntax:	DT()
Result:	Gives the step-size used for the simulation. This value is specified in the **Simulation Settings** form.

TL – Time length of simulation	
Syntax:	TL()
Result:	Gives the length in time for the simulation. This value is specified in the **Simulation Settings** form.

TE – End time for simulation	
Syntax:	TE()
Result:	Gives the End time for the simulation. TE() = TS() + TL().

10.2.6 Converter (Table look-up function)

The **Converter** is implemented as an element with its own button. Click on the converter button, which gives you a six-cornered figure. Double-click the *Converter to Define the data pairs* x_i, y_i. The data pairs should be separated by semicolon. (Space between each pair is inserted to make the table more readable.)



Name: [Converter1]

Definition: 0,6; 0.2,8; 0.4,14; 0.6,10; 0.8,7.3; 1.0,4

Converter – A table function (graph) that converts from input *X* to output *Y*	
Syntax:	Specify the X to Y conversions as a table in the Definition field.
Example:	You may want to use the empirically known Temperature as function of Time, Energy consumption as function of day of week or Fertility as function of Food supply in your model. Then you define the empirical function between an input X and an output Y. This relationship is expressed in tabular form. Consider for example the following relationship between X and Y: For example: 0,6; 0.2,8; 0.4,14; 0.6,10; 0.8,7.3; 1.0,4 (Shown in Figure 10.5.)

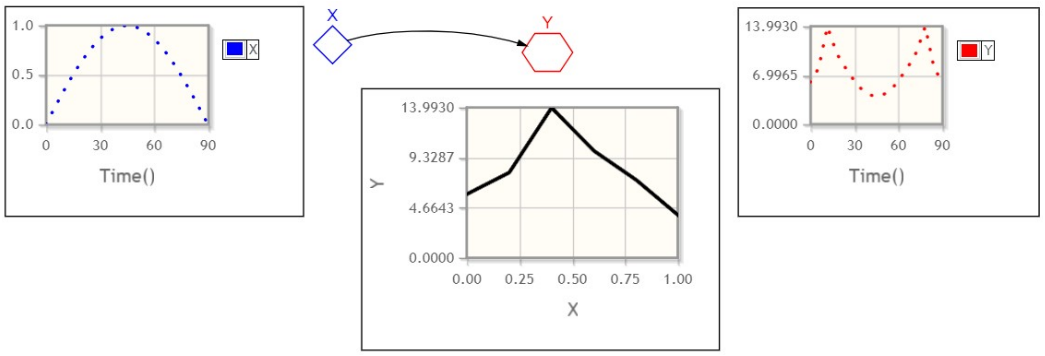


Figure 10.5. Empirical relation between input X and Output Y. Input to the converter is X(t)= Sin(Pi*T()/90), which is converted to Y(t) during 90 time units.

By using more points the table can describe the reality more accurately.

11. Macros

A **Macro** allows you to define model code that can be included in the model. Click the **Macros** menu to open a form to write the code.

A macro can be created in two forms (Example from the Insight Maker manual).

A single-line macro has the syntax: *Myfcn <- functional expression*.

```
  myFn(a, b, c) <- sin((a+b+c)/(a*b*c))
```

A multiline macro has the syntax:

```
Function myFcn(a, b, c)
   x <- (a+b+c)
   y <- (a*b*c)
   return sin(x/y)
End Function
```

Macros can also be used to define a variable or a constant that will be available in the model.

Example 11.1 Making a stochastic simulation run reproducible
To make a stochastic simulation model reproducible, you have to lock the seed for the random number generators in the model. Then the same sequences of random numbers will be generated for each simulation run. This can be done in the macro form by specifying the argument of the SetRandSeed function:

```
  SetRandSeed(17)
```

By changing the argument, you will get a new (reproducible) simulation run. ■

Example 11.2 Defining a Poisson flow in a simplified way
In Section 10.2.4 we added *PoFlow(lambda)* as a simpler to write function for *Poisson(Dt()*Lambda)/DT()*. If this function had not been available, you could have introduce it as a macro:
*PoFlow(Lambda) <- RandPoisson(Dt()*Lambda)/DT()*. ■

12. Practical tips

- Perhaps the most common trouble originates from *confusion about the units used*. Be very careful to specify your *time unit* as well as every other unit used. This can be done in plain text in the diagram. For example: 'Time in months, Rabbits in 1000's, Food in kg.'

- When you connect a Flow to a Stock, be sure that the connecting flow will be attached. The *Cloud* at the connected end will then disappear.

- *Name the elements before you define the formulas.* Say that you have the linked variables [Constant3] and [Stock1] linked to a Flow. You define the Flow as: [Constant3]*[Stock1]. Then you change [Constant3] to [c] and [Stock1] to [X]. The 'Linked elements:' will now be changed to [c] and [X] – but the 'Definition:' of the Flow will still be [Constant3]*[Stock1] and you will get an error message when running the model. (Of course you can then correct the Flow definition – but it is more convenient to name the elements before you make the definitions.

- The arrow of a flow points in the *positive* direction. This means that when a flow has a positive value it goes in the direction of the arrow, and when its value is negative it goes in the opposite direction. For example, the direction of the flow [Flow]=Sin(T()) will alternate over time.

- Note that a Flow can only be drawn in x- and y-directions. However, you can break the flow in right angles by clicking the *right* mouse button while drawing the flow. This is particularly useful when you have several flows between stocks.

- A link is represented by a so-called Bézier-curve that can be bent into a smooth curve. You can use this of aesthetic reasons – for example, you don't want to cross over other symbols. To do so, click on the link and use its handles to form the link.

- Sometimes the model graph will suffer from links crossing the model because the elements to be connected are far away. In such a case you may create a '*Ghost*' of an element at a proper place and draw a link from this ghost instead. (Remember that a Ghost can only have *outgoing* links and *no* flows.) To use a ghost, *mark* the element you want to ghost before clicking the **Ghost** button.

- Use colours to make the model more comprehensible.

- Also remember that the colour of an element will be used for the plot of this element in a Time diagram. Furthermore, In a **Time diagram**, **Stocks** will produce *solid lines*, **Flows** will produce *dashed lines* and the **other elements** will produce *dotted lines*.

13. References to deterministic CSS modelling

[1] [Forrester, J.W. (1961) *Industrial Dynamics*. Cambridge, MIT Press, MA. (The book where System Dynamics was introduced.)

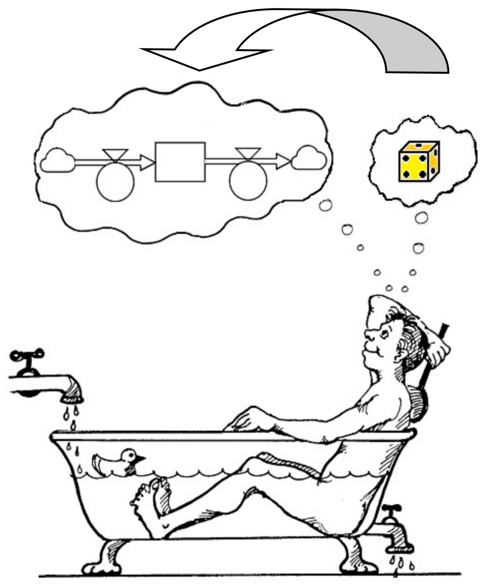
[2] Meadows, DH., and Wright D. (2008) *Thinking in systems: a primer*. White River Junction, Vt: Chelsea Green Pub. (An excellent first introduction to System Thinking and System Dynamics.)

[3] Fortmann-Roe, Scott. (2014) Insight Maker: A General-purpose tool for web-based modeling & simulation. Simulation Modelling Practice and Theory, 47, 28-45. http://www.sciencedirect.com/science/article/pii/S1569190X14000513

[4] The Manual for Insight Maker. (Here you find more features (e.g. functions) that are supported but not described in StochSim. Note however that only the System Dynamics part of Insight Maker is supported in StochSim.) https://insightmaker.com/book/export/html/40

[5] Home page for Insight Maker: https://insightmaker.com/

Part II. Stochastic modelling with StochSim



Extended purpose: To build *stochastic* and dynamic models and simulate their behaviours.

Extended worldview: The world consists of two kinds of fundamental elements: Flows and States. *However, lack of detailed information requires stochasticities to be included!*

14 Stochastic CSS Simulation

14.1 A historic note and the motivation for StochSim

Simulation languages on digital computers were introduced from around 1960. There are two main forms **Discrete Event Simulation** (DES) that is based on a micro approach where each individual or entity is to be represented and **Continuous System Simulation** (CSS of which StochSim is an example) where the individuals or entities are lumped into compartments (stocks). Here only the number or amount in a compartment is then represented.

However, when modelling the same system under study, the results from a DES and a CSS model were often inconsistent (i.e. not contradiction-free). During the remaining of the 20ies century this has been an issue of concern and discussion.

In a series of papers between 2000 and 2017, see references [6] to [12], how to handle different types of stochasticity, methods for handling queues and how to achieve sojourn times from different distributions in CSS were presented. It was also demonstrated that following some rules for CSS modelling regarding these issues, CSS and DES models will produce consistent results.

Furthermore, the conditions for a *deterministic* CSS model to produce unbiased results were investigated.

Then it was shown that different types of simulation (Agent-based, Entity-based, Compartment-based (i.e. CSS) and state-based (e.g. Markov) models) could all produce mutually consistent results. Furthermore, it was demonstrated how these types are related and how they can be translated into each other. Finally, the pros and cons of choosing one of these types instead of another were investigated.

Although, this knowledge can be applied using most CSS languages, stochasticity will require multiple simulation runs followed by a statistical analysis. Therefore, different modelling devices were also created, see [13] to [16]. These tools (presented in Chapter 16) are now implemented in StochSim. Especially StatRes (see Section 16.3) is in practice necessary for analysing multiple runs of a stochastic model.

14.2 What is stochasticity?

Every modelling study must have a well-defined purpose. The art of modelling is based on *only* describing what is relevant for this purpose. However, in modelling you often have to deal with *incomplete information*.

Lack of information (*uncertainty*) can be handled in two ways:

1) You can ignore that you have incomplete variation. For example unknown variations are replaced by e.g. an average, and hope that not too much damage is done, and that no one will notice. For example, if there is a dice involved, the possible outcomes 1, 2, 3, 4, 5, or 6 are always replaced by the average value (3.5). This is a common, stupid and dishonest way to operate.

2) You can handle lack of information by at least including the information you have, often in form of a probability density/distribution function (pdf). Then random numbers can be drawn from this pdf to produce stochasticity in the model. For example, a dice is described by a probability distribution function that has equal probabilities for the outcomes 1, 2, 3, 4, 5, and 6. A random number is then used to decide the outcome for each throw. This is the correct way of handling incomplete information. Then the model includes the information you actually have!

A model with stochasticity is a *stochastic* model; without stochasticity it is a *deterministic* model.

14.3 Five types of uncertainty

Five types of uncertainty about a system under study have to be handled in the model building process. These are:

- *Structural uncertainty* – incomplete knowledge about the structure of the system under study and about how the components interact.

- *Initial value uncertainty* – incomplete information about the initial conditions at 'time zero'.

- *Transition uncertainty* – incomplete information about when an event will happen (e.g. a customer arrives, a car will pass a bridge, a person gets sick).

- *Parameter uncertainty* – incomplete information about the value of a parameter or how it varies over time (e.g. when and how much will it rain).

- *Signal uncertainty* – incomplete knowledge about transferred information. A signal can be delayed or distorted.

Each of these uncertainties will affect the results of a model study. The more uncertain you are about structure, initial conditions, transitions, parameter values and signals, the more uncertain your results will become. A deterministic model would deny this fact!

While a deterministic model always will produce the same behaviour, a stochastic model will produce a different behaviour for each replication (simulation run). However, a stochastic model must be by executed a large number of times so that the uncertainty of the model behaviour can be studied by probability density/distribution functions of important *outcomes*. (Only if the important results of a deterministic model agree with those of a stochastic one, the deterministic model is consistent with the stochastic one. But event then, you will lose a lot of information about variations.)

One way to measure uncertainty in results is to use *confidence intervals*, which are easy to obtain by simulation. For example, a two-sided *95%* confidence interval is estimated in the following way: Just run the model, say, *1000* times. Sort the *1000* outcomes of a quantity you are interested in. Remove the *25* (*2.5%*) lowest and the *25* highest replications. The remaining *950* replications will span a two-sided *95%* confidence interval.

14.4 Uncertainties in a CSS model

A compartment-based model is constructed by *stocks*, *flows*, *parameters* (variables and constants), and *links*. There can be uncertainty related to each of these (in addition to the uncertainty of the structure used for arranging the just mentioned elements).

We use an example to demonstrate how and where different types of stochasticity can enter in a CSS model.

Example 14.1 *A classical SIR model*

A classical *SIR model* is an epidemic model composed of three stages: S (for Susceptible), I (for Infectious), and R (for Recovered). The following is assumed: A Susceptible person has the probability *p* to become infected by an Infectious one at each time unit. Furthermore, an Infectious person will recover and become immune after on average *T* time units.

A classical SIR model where the stages are represented by single compartments is shown in Figure 14.1. This model is used to discuss the five types of uncertainty mentioned above.

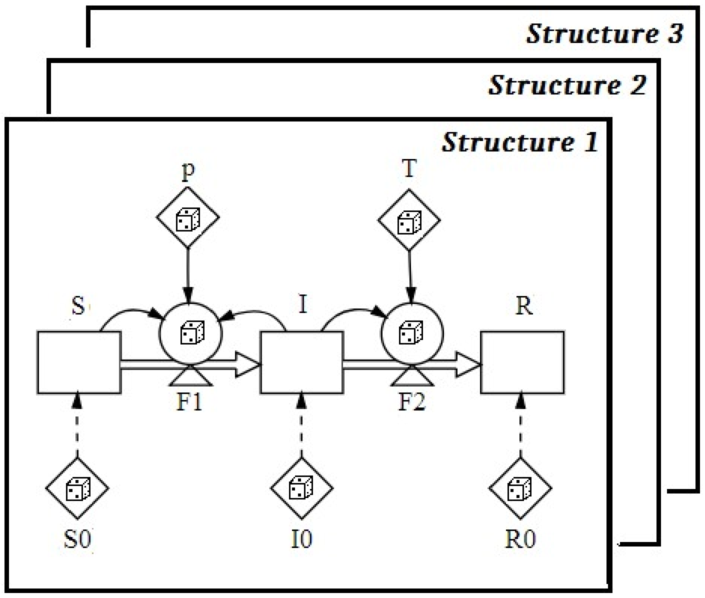


Figure 14.1. A classical SIR model where the stages *S*, *I* and *R* are represented by single compartments. The stocks are initiated to *S0*, *I0* and *R0*. The transition flows are denoted *F1* and *F2*, and the parameters are denoted *p* and *T*. Dice show where randomness to describe uncertainty can be located.

The five possible types of uncertainty: *Structural*, *Initial value*, *Transition*, *Parameter*, and *Signal uncertainties* will all be recognised for this example. In a compartment-based model the *structural uncertainty* affects how to describe the system under study in terms of an appropriate *model structure* of stocks, flows, parameters and links, while the other four uncertainties appear at different elements *within* the chosen model. *Initial uncertainty* relates to stocks, *Transition uncertainty* relates to flows, *Parameter uncertainty* relates to variables or constants, and *Signal uncertainty* relates to (information) links.

1. Structural uncertainty

The description of the SIR model is not complete. Especially, the time in the Infectious stage (the so-called *sojourn time*) is on average *T* time units. But some persons will recover sooner and other will require a longer time. By modelling the Infectious stage by a single compartment, we have implicitly assumed an exponential sojourn-time distribution. (Other assumptions about this distribution can be realised by using several compartments in parallel and/or series.)

Another structural possibility is that an infected person does not immediately become infectious. The exposed person may require a latent period before he enters the Infectious stage. This can be accomplished by including an Exposed (E) stage between the Susceptible and the Infectious stages (giving a so-called SEIR model). Perhaps, also an unknown number of the population was already immune, i.e. belonging to R instead of S?

2. Initial value uncertainty

The information about the number of persons in the three compartments at time zero may be approximate. Say that we know that the population in a village is $N=1000$ persons. Three persons from this village returns from a trip abroad bringing home a new disease, but we don't know whether 1, 2 or all 3 of them were infectious. The *initial value* for the stock $I(0)$ is then 1, 2 or 3 according to some probability distribution.

3. Transition uncertainty

There is almost never enough information to decide the exact event times for a transition. Therefore, *transition uncertainties* in the flows $F1$ and $F2$ must be included in the description of the flows, using available statistical information for the transitions.

In a population model (e.g. a SIR model), transition uncertainty is usually the most fundamental and important to handle. It is easy and straightforward to implement and it will change the nature of the model from a continuous mass of a population to distinct individuals, which in turn can have a number of consequences.

4. Parameter uncertainty

Unexplained irregularly varying values of the parameters p and T may motivate the description of *parameter stochasticity* to generate realistic inputs from the parameters. In this case there is no given form to describe this uncertainty. Only statistical knowledge about the parameter variations or uncertainty of a parameter value can guide you here.

5. Signal uncertainty

Before we discuss signal uncertainty we have to clarify that information links in a model can be of two different types:

a. *Artificial link* – a technical concept to communicate logic between artificially separated model parts. For example when a radioactive atom decays, there is one radioactive atom less in the system under study. In a compartment-based model this is accomplished by a construction of a stock and a physical outflow. An artificial link from the stock to the valve regulating the outflow has here to be included. Its only function is to transfer information about the content (value) of the stock to the flow equation. There is no counterpart to such a link in the real system under study, and *no uncertainty or delay can be involved.*

b. *Signal link* – a description of a real information link that communicates information in the system under study. This communicated information we call '*signal*'. A signal can be *distorted* and/or *delayed*.

In this example there are no signals. However, we can assume that there is a responsible authority for epidemics that collects information about the number of Infectious persons (which takes time and is never exact). This authority may issues recommendations about avoiding contacts and to use sanitary measures, which may affect the infection rate, F1, (when the message later on is received by some fraction of the population).

Signals require time to reach the receiver that may vary in an unpredictable way (the information may be sent by post, it may arrive during the weekend but noticed first on the following Monday, etc.), and the information may be distorted because of various reasons (imperfect information, typing error, distortion of the signal, misinterpretation, not reaching all subjects, etc.).

The length of a delay can be generated by drawing a random number from a proper distribution and the distortion may be treated as stochastic parameter for the information about the number of Infectious. In this way, uncertainty about signals can be handled.

Initial uncertainty, *Transition uncertainty*, *Parameter uncertainty* and *Signal uncertainty* are all treated in a similar way. First you describe the uncertainty by a statistical distribution. Then you draw a random number from this distribution.

For example, assume that you estimate the probability of the *I*-stock in the SIR model initially being *I(0=1* infectious person to *60*%, *I(0)=2* to *30*% and *I(0)=3* to *10*%. Then you can draw a uniformly distributed random number between *0* and *1* and use this distribution to interpret the outcome. If $outcome \leq 0.6$ then *I(0)=1*, else if $outcome \leq 0.6+0.3$ then *I(0)=2*, else *I(0)=3*. ■

After this introduction we will focus on how to construct transition uncertainty.

14.5 Implementing transition uncertainty in a CSS model

A *population model* describes the development of a population of persons, cars, rabbits, molecules, radioactive atoms, etc. The number of a population or subpopulation is integer.

In a compartment-based description every stock must contain an integer number. A continuous flow, where the flow rate is expressed as a *fraction* of the content in a stock (as in a deterministic model) is no longer valid. For example, a real system of decaying radioactive atoms will run out of radioactive atoms in a finite time, while a *deterministic* model of this system removes a certain fraction of remaining radioactive atoms per time step why the number of radioactive atoms never will reach zero in the model.

The way to make the model correct is the following:

1) Initiate the stocks to integer values.
2) The in- and outflows to and from a stock consist of sequences of events, where integer numbers of items arrive or depart during each time-step.

Transition stochasticity is here exemplified by a simple example of *N=50* radioactive atoms decaying over time with the time constant *T*, see Example 15.1. A *deterministic* model is:

N(t+DT) = N(t) – DT·F(t) (where *N(0)=60*)
F(t) = N(t)/T

Here *DT·F(t)* is the number of decays during the time-step *DT*. This amount 'flows' out from the compartment *N* at each time-step.

In the *stochastic* case *DT·F(t)* is instead the *expected* number of decays during the time-step *DT*, and we know that this number must be integer. The probability theory tells us that the number of events (here decays) during a short time period (here *DT*) is *Poisson distributed*. This means that *DT·F(t)* is replaced by Po[*DT·F(t)*]; where Po stands for the Poisson distributed random number function that is included in almost every simulation language. (In StochSim this function is named RandPoisson(·).)

However, there is a minor complication. The modelling syntax is: *F(t) = N(t)/T* (without the *DT*). But the Poisson distribution tells about the number of events *during a time period* (here DT). Therefore, we must multiply left and right hand sides of *F(t) = N(t)/T* by DT to obtain *DT·F(t) = DT·N(t)/T*, which in the stochastic case becomes: *DT·F(t)*] = Po[*DT·N(t)/T*]. Finally, dividing left and right hands of the equation by *DT* gives: *F(t)*] = Po[*DT·N(t)/T*]/*DT*.

The stochastic model thus becomes:

N(t+DT) = N(t) – DT·F(t) (where *N(0)=60*)
F(t) = Po[*DT·N(t)/T*]/*DT*

This is the general form for transition stochasticity.

Notice that as long as *N* is very large, the difference between the results from a deterministic and a stochastic model is negligible.

WARNING: In a *deterministic* case, you may *add* or *subtract* flow equations:
Say that [Flow1]=[c1]*[X1] and [Flow2]=[c2]*[X1].
Then [NetFlow]=[c1]*[X1] + [c2]*[X1] or
[NetFlow]=[c1]*[X1] – [c2]*[X1].

For the corresponding *stochastic* case this is only valid for *addition*:
[NetFlow]=PoFlow(DT()*[c1]*[X1] + DT()*[c2]*[X1])/DT().
However, *subtraction* is not valid. Here you *cannot* subtract the arguments. You must then *stay with* two Poisson terms:
[NetFlow]=PoFlow(DT()*[c1]*[X1])/DT – PoFlow(DT()* [c2]*[X1])/DT().

A simple argument for this is: Say that c1=5 and c2=4. But the *random outcome* may occasionally be larger for the second term giving a negative NetFlow. However, PoFlow(DT()*5*[X1] – DT()*4*[X1])/DT() = PoFlow(DT()*1*[X1])/DT() can never be negative, because Po(negative value)=0.

14.6 What can happen if you ignore stochasticity?

There is a widespread misconception that you should ignore information you don't know exactly and e.g. replace it with an average value, ending up with a deterministic model. This is a stupid, dishonest and dangerous attitude that often leads to biased results, exclusion of phenomena, erroneous conclusions and no insight in the precision of the estimates.

A dynamic *and stochastic* model should be based on all relevant information – even though this information may not be complete!

When a population system under study is modelled as a *deterministic* population model, various types of distortions may occur. For example:

- Information about *irregular variations*, *risks*, *extremes*, *correlations*, the possibility to calculate *confidence intervals*, etc. are all lost.
- The elimination of transition stochasticity may introduce *bias* in the average behaviour and result in *biased* estimates.
- Important *phenomena*, such as extinction of a population or *elimination of an epidemic*, which can happen in the system at study and is reflected in stochastic modelling, will be lost.
- A deterministic model produces *categorical answers*, e.g. that a force *X* will win over a force *Y*, whereas a stochastic model will give the probabilities of *X* and *Y* as winners.
- *Oscillations* might disappear in a deterministic model although they should be excited by transition stochasticity.
- In a deterministic model, the lack of transition stochasticity may *prevent queues* from building up.
- *The time* until a specific event occurs will (correctly) vary between replications in a stochastic population model, but will be the same in a deterministic one. A deterministic model may then introduce bias in the estimates. Furthermore, in a deterministic model this time can even erroneously become infinitely long, even when it should be finite.

In short, using a deterministic population model will also mean that we only are aiming for average estimates – loosing all insight in variations and extreme behaviours. Unless this average will equal the average value obtained by a corresponding discrete and stochastic model, the use of a deterministic model will produce biased results and prevent interesting phenomena to be displayed.

14.7 A warning example – Comparing the results from a deterministic and a stochastic SIR model

Assume the following for the classical SIR model, presented in Section 14.4 above. The initial population *N=S+I+R* consists of *1000* susceptible persons, a single infectious person and no recovered persons. The infection risk parameter is *p=0.0003* per person and time unit and the sojourn time in the infectious stage is *T=4* time units.

10 000 replications of the (transition) stochastic SIR model to study the cumulated number of susceptible individuals being infected, give an ensemble of results with the average value of *53.1* persons [*95% confidence interval: 50.8 – 55.5* persons]. However, the corresponding deterministic model produces *318.5* persons. An error of about *500* percent was introduced by dropping the transition stochasticity.

The reason why the deterministic model distorts the results is because the initial infectious subpopulation *I(0)=1* is small. There is then a large chance that the single infectious person will recover before infecting another susceptible, so that no epidemic will occur.

15. A stochastic StochSim model

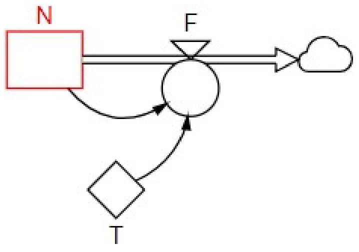
Example 15.1. Radioactive decay
A specimen contains *N=60* radioactive atoms will decay with a time constant *T=10* minutes is studied during *30* minutes.

A model can be mathematically described by the equations:

N(t+DT) = N(t) - DT·F(t)	(where N(0) = 60)
F(t) = Po[DT·N(t)/T]/DT	(In a deterministic model: F(t) = N(t)/T.)

Po[.] stands for a Poisson distributed random sample.

A StochSim model and a replication are shown below.



Model equations
N: 60
T: 10
F: RandPoisson(DT()*[N]/[T])/DT()
Alternatively:
F: PoFlow([N]/[T])

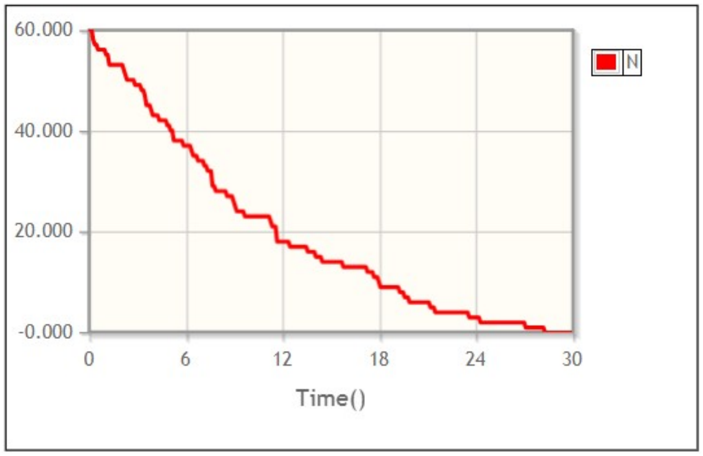


Figure 15.1. A stochastic model of radioactive decay and a possible outcome.

Note that the definition of the Flow [F] as 'RandPoisson(DT()*[N]/[T])/DT()' can be simplified by using the function PoFlow as: 'PoFlow([N]/[T])'.

16. Tools in StochSim

StochSim contains four important tools:, **Optim**, **Sensi, StatRes** and **ParmVar** (for Optimisation, Sensitivity analysis, Statistical analysis & result presentation, and Parameter estimation variations). These tools are found in the **Tools** menu. The selected tool will show up in a separate window to the right on the screen, and it works on the model displayed in the Model window. Each of these four tools also has its own documentation. Here we only indicate what they are used for.

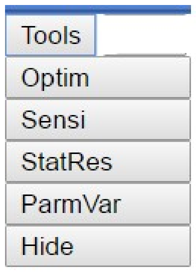


Figure 16.1. The Tools menu.

16.1. The Optim tool

Optim (Optimisation) is a simplex optimiser for finding the optimum (maximum or minimum) of an objective function defined in a *deterministic* model with respect to a number of model parameters.

An optimiser has two main purposes in a deterministic model. First, an optimiser can be used *to find the optimal strategy* for a model, e.g. minimise costs or maximise benefits. To do this, an *objective function* is defined in the model as function of a number of model quantities. Then the optimiser systematically varies the values of specified parameters to find the optimal set of parameter values.

Second, it can be used for *parameter estimation*. Then you construct a function that measures *the difference* between the system's and model's behaviours. This function is then *minimised* with respect to a set of parameters. The set of parameter estimates obtained are those which produce the model behaviour that best reproduces the behaviour of the system under study. Parameter estimation is often a necessary part of the construction of a model.

Optimisation is easy to perform a on deterministic model – and much trickier on a stochastic one.

The simplex optimiser is reasonable fast and has the advantage that you easy can include boundaries in the parameter space. For more information, see the Optim manual [13].

16.2. The Sensi tool

Sensi (Sensitivity Analysis) is a tool for sensitivity analysis. With this tool you can study the effect on an outcome from changing a parameter or initial value. For example you may study how the price of a commodity will affect the revenue of a company. For more information, see the Sensi manual [14].

16.3. The StatRes tool

The price to pay for using a stochastic simulation model is that many replications are required, which must be followed by a statistical analysis.

StatRes (Statistical Results) orders a model to be run a specified number of times. Then a statistical analysis of specified outcome quantities (Stocks, Flows or Variables) is performed. The *Average*, *Standard deviation*, *Min*, *Max*, and *Percentile*s for the performed number of replications are presented for each specified outcome quantity. You can also present the outcome quantities from the replications in a tabular form, or as a histogram. Further you can plot two outcomes in an xy-plot and also obtain an estimate of the correlation between the two quantities. For more information, see the StatRes manual [15].

16.4. The ParmVar tool

ParmVar (Parameter Variations) is a tool for assessing the accuracy of previously estimated parameters. After parameter estimation (with e.g. **Optim**) an important question remains: How *accurately* are the set of parameters values estimated? This question can be addressed with ParmVar.

Parameter estimation is tricky to perform on a *stochastic* model. It requires a large number of replication for each tested set of parameter values (as compared to a single replication in the deterministic case). However, for two important classes of models the expected outcome from a stochastic model is the same as the outcome from a corresponding deterministic one. Then you can find the optimal set of parameter values with **Optim**.

This is a more advanced tool that usually will not be used in an introductory course. For more information, see the ParmVar manual [16].

16.5. Hiding the tool

The Hide option hides a displayed tool and lets the Model window occupy the whole width of the screen.

17. References to stochastic CSS modelling

[6] Gustafsson, L. (2000) Poisson Simulation – A Method for Generating Stochastic Variations in Continuous System Simulation. *Simulation*, **74**, 264-274. http://dx.doi.org/10.1177/003754970007400505

[7] Gustafsson, L. (2003) Poisson Simulation as an Extension of Continuous System Simulation for the Modeling of Queuing Systems. *Simulation*, **79**, 528-541. http://dx.doi.org/10.1177/003759703040234

[8] Gustafsson, L. and Sternad, M. (2007) Bringing Consistency to Simulation of Population Models-Poisson Simulation as a Bridge between Micro and Macro Simulation. *Mathematical Biosciences*, **209**, 361-385. http://dx.doi.org/10.1016/j.mbs.2007.02.004

[9] Gustafsson, L. and Sternad, M. (2010) Consistent Micro, Macro and State-Based Modelling. *Mathematical Biosciences*, **225**, 94-107. http://dx.doi.org/10.1016/j.mbs.2010.02.003

[10] Gustafsson, L. and Sternad, M. (2013) When Can a Deterministic Model of a Population System Reveal What Will Happen on Average? *Mathematical Biosciences*, **243**, 28-45. http://dx.doi.org/10.1016/j.mbs.2013.01.006

[11] Gustafsson, L. and Sternad, M. (2016), A guide to population modelling for simulation. OJMSi, 4, 55-92. http://file.scirp.org/pdf/OJMSi_2016042717425486.pdf

[12]] Gustafsson, L., Sternad, M. and Gustafsson E. (2017), The full potential of Continuous System Simulation modelling, (2017) OJMSi, 5, 253-299. http://www.scirp.org/JOURNAL/PaperInformation.aspx?PaperID=80104

[13] Gustafsson, L., and Gustafsson E. (2017) Optim – An optimiser for deterministic StochSim models. https://stochsd.sourceforge.io/homepage/docs/Optim.pdf

[14] Gustafsson, L., and Gustafsson E. (2017) Sensi – A sensitivity analyser for StochSim models. https://stochsd.sourceforge.io/homepage/docs/Sensi.pdf

[15] Gustafsson, L., and Gustafsson E. (2017) StatRes – A tool for statistical analysis of stochastic StochSim models. https://stochsd.sourceforge.io/homepage/docs/StatRes.pdf

[16] Gustafsson, L., and Gustafsson E. (2017) ParmVar – A tool for studying the variation of parameter estimates in StochSim models, 2017.ParmVar. https://stochsd.sourceforge.io/homepage/docs/ParmVar.pdf

Appendix A. Installation of StochSim

StochSim is available in two versions: **StochSim Desktop** and **StochSim Web**.

A.1 StochSim Desktop

System requirements
StochSim Desktop can be used under the operative systems Windows, Mac OS X or GNU/Linux.

Installation instructions
The Desktop version is downloaded from StochSim's homepage and runs as a normal program. Download it from StochSim's home page:
https://stochsd.sourceforge.io/homepage

Find the *StochSim.zip* file and unzip it into a folder of your choice on your computer.

A.2 StochSim Web

System requirements
StochSim Web can be used under any operative system that can run a modern web browser. We recommend Mozilla Firefox and Google Chrome. (StochSim does not work on Internet Explorer.)

Running instructions:
Go to https://stochsd.sourceforge.io/software in Firefox or Google Chrome.

Appendix B. License and Responsibility

B1. License

StochSim is based on the open source part of Insight Maker. StochSim's source code (including the open source part of Insight Maker) is released under a custom license called *Insight Maker Public License*, which is based on the Affero GPL. Insight Maker's Public License is available at: https://insightmaker.com/impl.

Insight Maker Public License thus covers all StochSim's JavaScript, HTML and CSS code for StochSim. The original non-open source code in Insight Maker, such as ExtJS and mxGraph are completely eliminated and replaced in StochSim.

B2. Responsibility

The user is fully responsible for the use of StochSim. The producer and the supplier of this code take *no* responsibility for the use or functioning of StochSim and its tools.