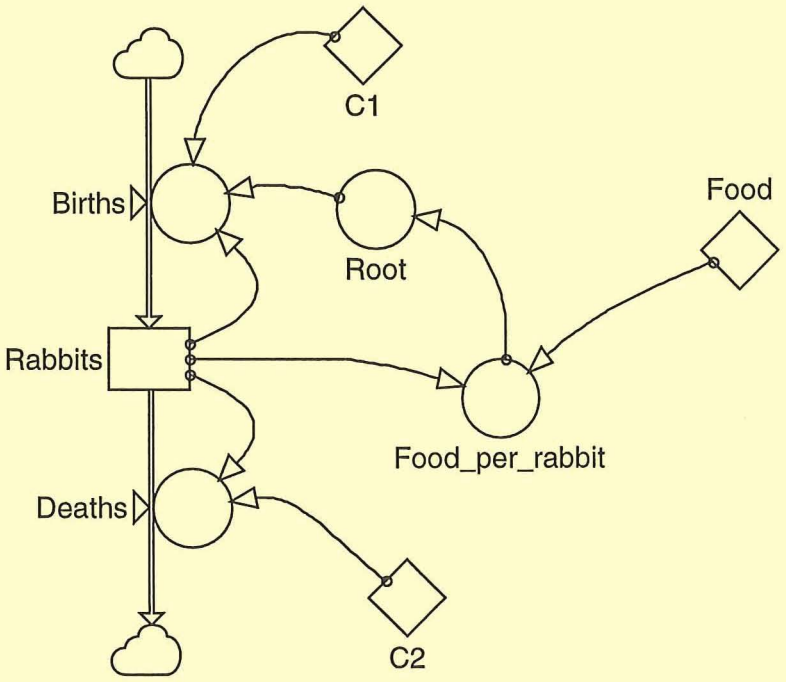
Systems Analysis Group, 1998
Uppsala University

# Powersim

# -

# A short introduction




Annica Karlsson
Thomas Persson
Oktober 1998

# 1 Introduction

Powersim is a tool for modelling and simulation of dynamic systems. It can be used to study time continuous progress in a great number of areas, for example biology, economics, physics and ecology. The modelling is done by constructing a Powersim diagram. This is done by choosing from the set of defined graphical symbols, and placing them on suitable places. Elements influencing each other are then connected with arrows. See the example of coffee cooling in a cup given in figure 1.

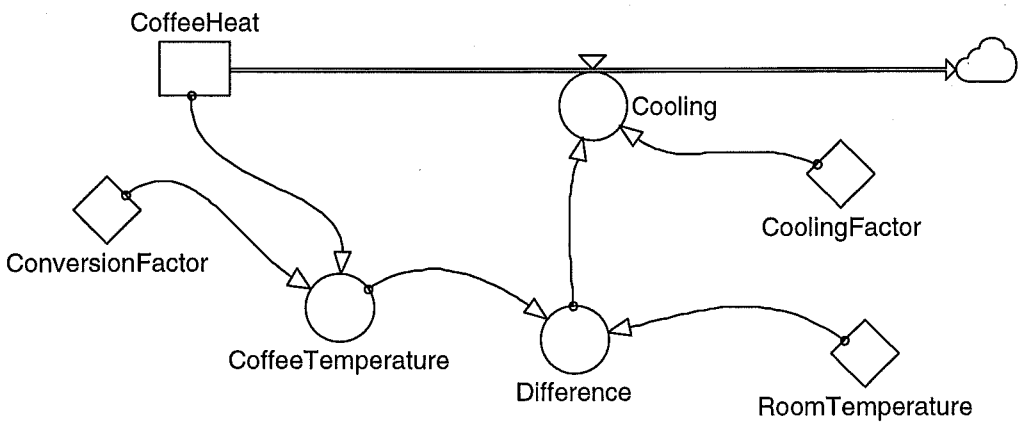



**Figure A.** Powersim diagram. Coffee cooling in a cup.

Powersim is easy to understand and use even for people without great knowledge in mathematics, programming and simulation. The integration is based on a "bath tub analogy" which doesn't demand knowledge about differential calculus.

The state L (**L**evel) in figure 2 is affected by an inflow R_in (**R**ate) and an outflow R_out. This corresponds to a bath tub where the water level is affected by two taps, one that runs water into the bath and one that lets water out of the bath.

Note that the state L is not entirely determined by the flows, but also by the initial value $L_0$. Note also that the state does not depend on the current flows, but on the accumulated flows from the past.

Mathematically is L determined by the equation

$$\begin{cases} \frac{dL}{dt} = R\_in(t) - R\_out(t) \\ L(0) = L_0 \end{cases}$$

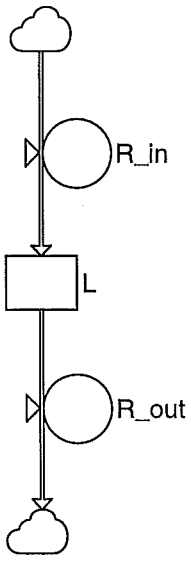



**Figure B.** Bath tub analogy.

i.e.

$$L(t)=L_0+\int_0^t (R\_in(t)-R\_out(t))\,dt$$

In Powersim first-order systems of differential equations are modelled and simulated. The differential equations are often coupled, i.e. the states are dependent of each other.

Powersim includes many dynamic objects for input and output of data to and from the model. Examples of these objects are time graphs, tables, sliders and bars. There are also some static objects to help presenting the model. These are text objects, frame objects, picture objects and line objects.

# 2 The simulation tool Powersim

## 2.1 Powersim diagram

The Powersim diagram is a flow diagram that shows the model divided into states and flows. In addition there are also constants and auxiliaries. The diagram is automatically translated into equations to give an textual representation of the model.

Some of the most common symbols are shown below in figure 3.

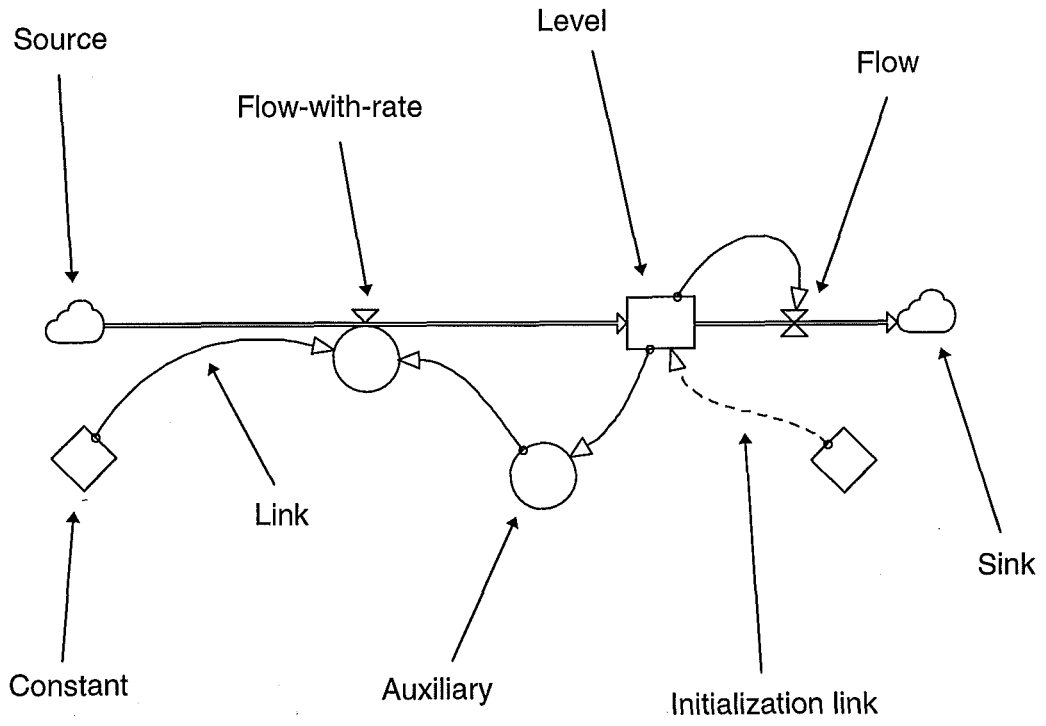



**Figure C.** The most common symbols in Powersim.

Below is a brief comment to each of the symbols in figure 3:

| | | |
|---|---|---|
| Level | - | State variable. For example amount of water ($m^3$), energy (J), populations (number of individuals). |
| Flow | - | Transport between levels. For example water flow ($m^3/s$), births (individuals/s). |
| Source/sink | - | Origin or destination of a flow. The real source/sink is outside the system and is therefore not described in the model. |
| Constant | - | Constant that affects the system. |

| | | |
|---|---|---|
| Auxiliary | - | Evaluate mathematical expressions. |
| Link | - | Information. Describes which variables that influences each other. |
| Initialization link | - | A link to a level is considered to be an initialization link, i.e. a link that shows what determines the initial value of the level. |

Auxiliaries are often used to control flows, but they can also be used to evaluate useful or interesting quantities or to make the diagram clearer. An auxiliary can have any number of inputs, while a **flow** can have one input. Therefore is the **flow-with-rate** symbol often used. The flow-with-rate actually consists of an auxiliary linked to a flow. Since there is no way of knowing from a diagram how he variables depend on each other, it is the custom to let all linked variables (input variables) to a flow-with-rate be multiplied.

An auxiliary is defined by an equation. Besides the arithmetic operations ( +, –, *, / ) there are a number of functions that can be used. They are presented in chapter 3. Some types of functions are very common and have their own Powersim symbols so that they can be distinguished from each other. See figure 4 below.

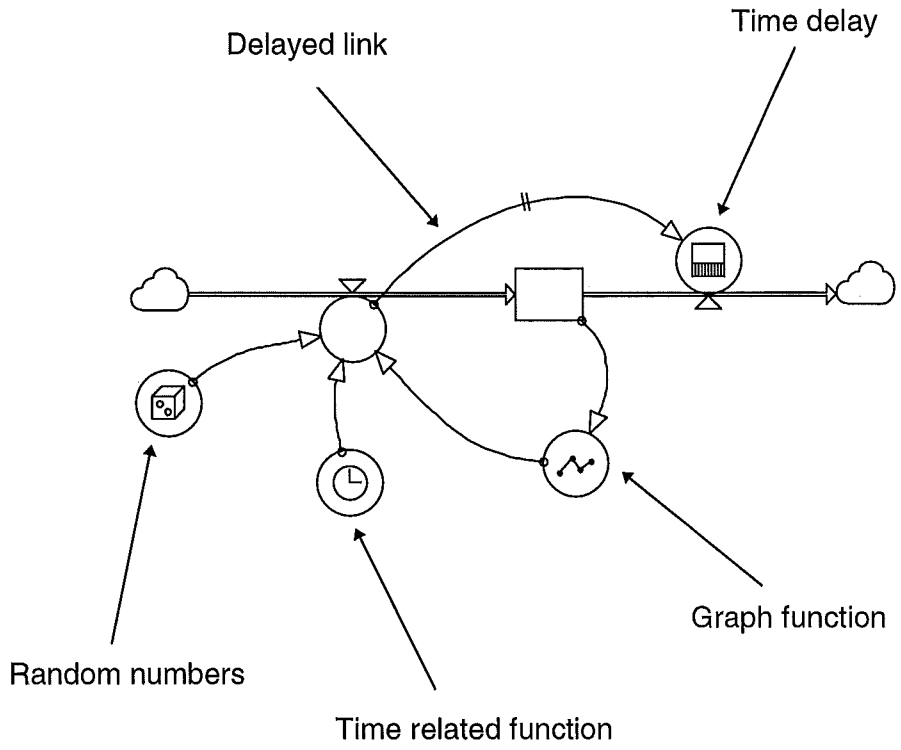



**Figure D.** The rest of the Powersim symbols.

**Example**:

We wish to study the progress of a rabbit population on an isolated island in the sea where 10 rabbits are introduced. They eat grass, increase and die of old age.

Studies of this rabbit society has shown that the food available is constant and is estimated to 100 kg per month. It has also been shown that the number of rabbits born per month are proportional to the size of the population and to the square root of the amount of food per rabbit. The proportional 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 that 5% of the population dies every month.

The causal-loop diagram for the model is shown in figure 5 below.

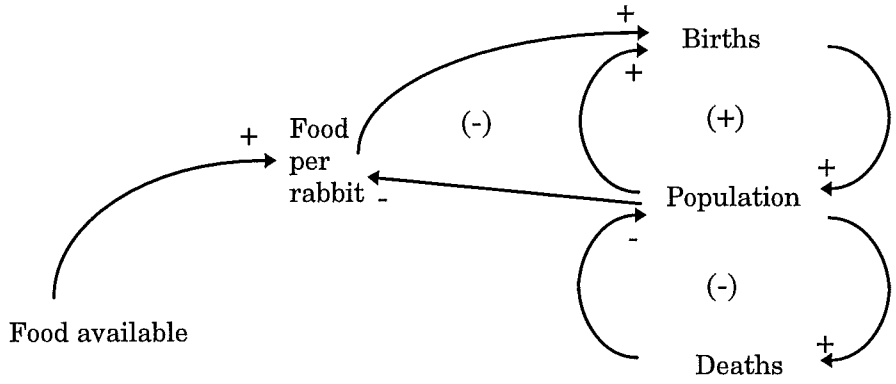



**Figure E.** Causal-loop diagram of the rabbit model.

If we introduce the proportional constants C1 = 0.2 and C2 = 0.05 the Powersim diagram for this model will be:

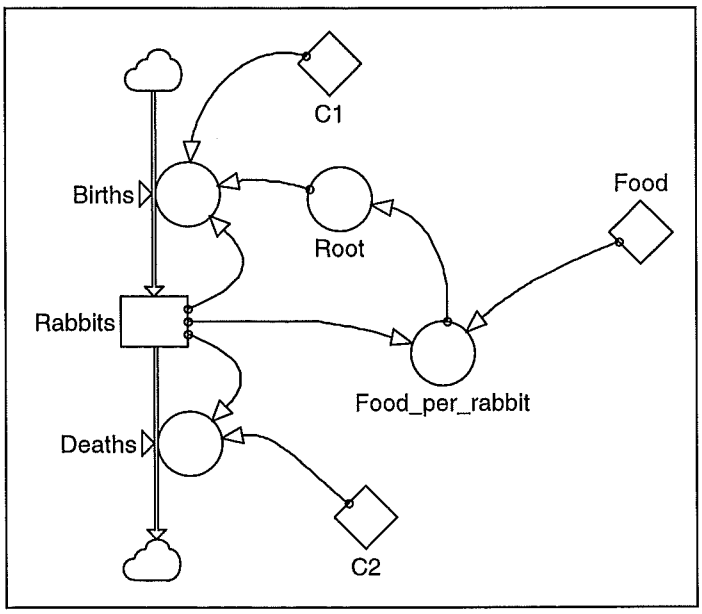



**Figure F.** Powersim diagram of the rabbit model.

## 2.2 Powersim equations

Powersim automatically generates Powersim equations from the Powersim diagram. These equations give an textual representation of the model. By studying the equations a more detailed view of the model is obtained than by just looking at the diagram. From the Powersim equations you can for example read off the initial values of the levels, the values of the constants and how the auxiliaries are defined.

How are the auxiliaries, constants, initial values, etc. defined? When you double click on a symbol the dialog box ***Define Variable*** appears. There you can define the variable corresponding to the symbol you clicked on. See chapter 4.

**Example, continued**:

The Powersim equations for the rabbit model are

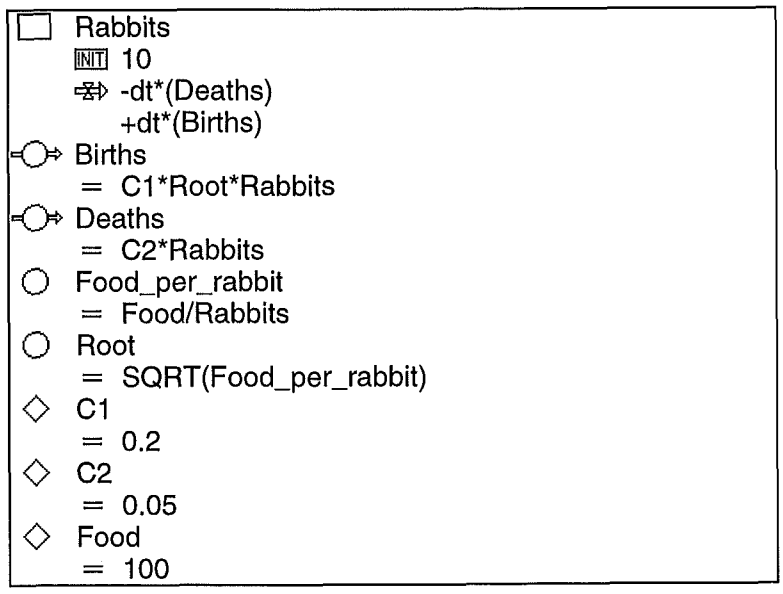
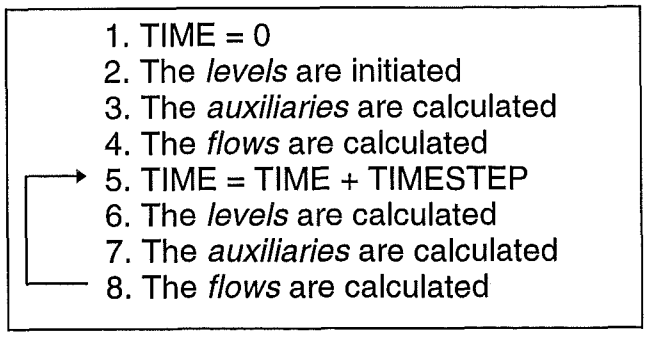



**Figure G.** Powersim equations for the rabbit model.

What happens during a simulation? Powersim has in principle the following working procedure (TIMESTEP is the step length):



## 2.3 Modelling in Powersim

Since Powersim is a graphical tool, one should utilize the possibility of having diagrams that are well structured and easy to understand. The Powersim diagram should therefore be as clear as possible to be able to present the model. There are a few rules to consider when constructing a diagram:

- Inflows to a state should be used when something is brought to a level, for example water, individuals or energy. Outflows from a level should be used when something is removed from the level. Use therefore inflows at births, energy influx, water inflow etc., and outflows at deaths, energy loss, leakage etc. See for example the rabbit model in figure 6 where Births is modelled as an inflow and Deaths as an outflow.

- Use auxiliaries to simplify the understanding of the diagram. An example is the variable Root in figure 6.

- Constants that affects the system should be shown explicitly and should not be hidden in the Powersim equations.

- The names for the levels, auxiliaries, flows etc should be chosen carefully.

- All the links into a flow regulator should be multiplied. See for example figure 6 where the links from C1, Root and Rabbits into Births indicate that Births is defined by Births=C1*Root*Rabbits.

**Example of a very bad modelling**:

The rabbit example can have this Powersim diagram and equations:

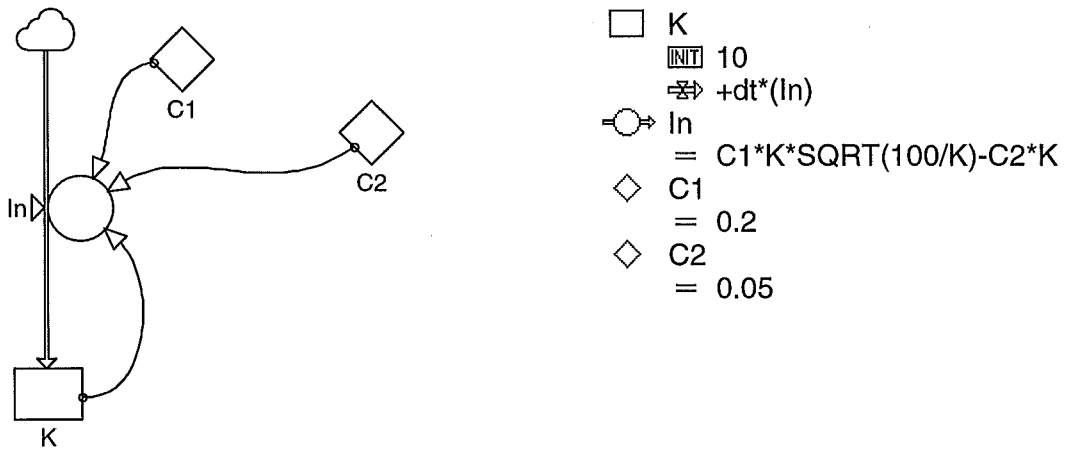



**Figure 8.** Example of bad modelling.

The simulation result will be exactly the same with this solution as with the solution presented earlier. *However*, the diagram and the equations are very difficult to understand! You can't understand the model from this diagram. The Powersim equations doesn't give you any help either. What does the term 100/K mean? What is -C2*K? What is K?

# 3 Functions

In Powersim there are many library functions. The most common functions are presented in this chapter. A more detailed presentation is given after this summary.

*Mathematical functions*
- A + B, A- B, A * B, A / B

- SQRT(X)
- A ^ B

- SIN(X) ARCSIN(X)
- COS(X) ARCCOS(X)
- TAN(X) ARCTAN(X)

- LOG(X, Base)
- LN(X) EXP(X)

- PI

- ABS(X)

*Statistical functions*
- MIN($X_1$, $X_2$, ..., $X_N$)
- MAX($X_1$, $X_2$, ..., $X_N$)

*Random functions*
- EXPRND(Mean, Seed)
- NORMAL(Mean, Stdv, Seed)
- POISSON(Mean, Seed)
- RANDOM(Min, Max, Seed)

*Time related functions*
- PULSE(Volume, $T_0$, Interval)
- RAMP(Slope, $T_0$)
- STEP(Amplitude, $T_0$)
- TIME
- TIMESTEP

*Graph functions*
- GRAPH(In, $X_1$, dx, Y)
- GRAPHCURVE(In, $X_1$, dx, Y)
- GRAPHLINAS(In, $X_1$, dx, Y)
- GRAPHSTEP(In, $X_1$, dx, Y)

*Delay functions*
- DELAYINF(In, Delay, Order, Init)
- DELAYMTR(In, Delay, Order, Init)
- DELAYPPL(In, Delay, Init)

*Rounding*
- CEIL(X)
- FLOOR(X)
- ROUND(X)

*Logical functions*
- IF(Condition, Expression1, Expression2)
- A < B, A > B, A <= B, A >= B, A = B, A <> B

**Matemathical functions**

| Syntax | Function | Comment |
|---|---|---|
| SQRT(X) | $\sqrt{X}$ | |
| A ^ B | $A^B$ | |
| SIN(X) | sin X | X in radians |
| LOG(X, base) | $\log_{base} X$ | |
| LOG(X) | $\log_{10} X$ | The 10-logarithm is default |
| LN(X) | ln X | |
| EXP(X) | $e^x$ | |
| PI | $\pi$ | |
| ABS(X) | abs(X) | The absolute value of X |

**Statistical functions**

| MAX - Maximum | |
|---|---|
| Syntax: | MAX($X_1$, $X_2$, ..., $X_N$) |
| Result: | Returns the maximum of the arguments. |

| MIN - Minimum | |
|---|---|
| Syntax: | MIN($X_1$, $X_2$, ..., $X_N$) |
| Result: | Returns the minimum of the arguments. |

## Random functions

EXPRND - Exponentially distributed random numbers

Syntax: EXPRND(Mean, Seed)

Result: Generates exponentially distributed random numbers with a mean of *Mean*.

Note: *Seed* is the initialization seed of the random number generator. If no seed is given, *Seed* will get a random value.

Example: EXPRND(2) gives exponentially distributed random numbers with a mean of 2.

NORMAL - Normally distributed random numbers

Syntax: NORMAL(Mean, Stdv, Seed)

Result: Generates normally distributed random numbers with a mean of *Mean* and a standard deviation of *Stdv*.

Note: See EXPRND.

Example: NORMAL(3, 0.5) gives normally distributed random numbers with a mean of 3 and a standard deviation of 0.5.

POISSON - Poisson distributed random numbers

Syntax: POISSON(Mean, Seed)

Result: Generates Poisson distributed random numbers with a mean of *Mean*.

Note: See EXPRND.

Example: POISSON(2) gives Poisson distributed random numbers with a mean of 2.

RANDOM - Uniformly distributed random numbers

Syntax: RANDOM(Min, Max, Seed)

Result: Generates uniformly distributed random numbers between *Min* and *Max*.

Note: See EXPRND.

Example: RANDOM(-1, 1) gives uniformly distributed random numbers between –1 and +1.

## Time related functions

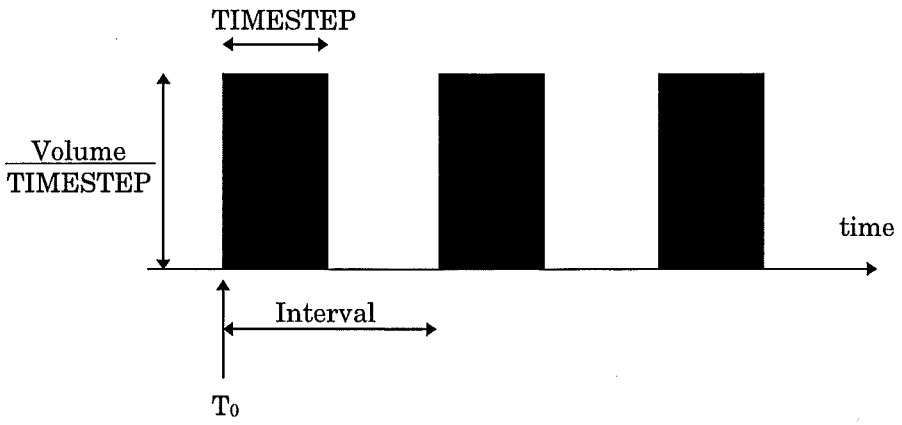
PULSE - Pulse generation

Syntax: PULSE(Volume, $T_0$, Interval)

Result: Generates pulses with the amplitude *Volume*/TIMESTEP and the pulse width TIMESTEP. *To* is the time of the first pulse and *Interval* is the interval between the pulses.

Note: TIMESTEP is a function that returns the step length of the simulation. See TIMESTEP.

If only one pulse is to be generated, then the interval can be set to a number larger than the length of the simulation, e.g. PULSE(Volume, $T_0$, 999999).



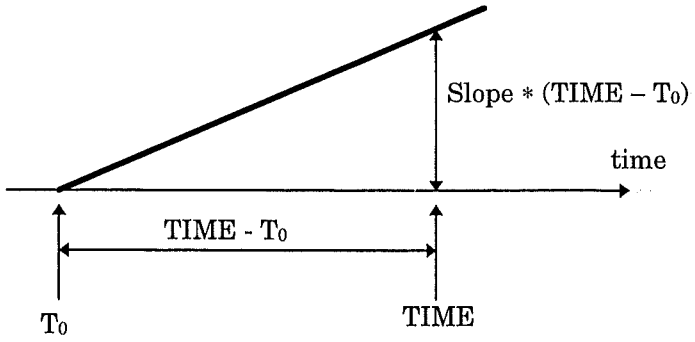

Example: Let TIMESTEP = 0.2. PULSE(1,4, 2) then generates a train of pulses with the amplitude 1/0.2 = 5 and the pulse width 0.2. The first pulse appears at time 4. The interval between the pulses is 2.

RAMP - Ramp generation

Syntax: RAMP(Slope, $T_0$)

Result: Generates a ramp with the slope *Slope* that starts at time *To*. It returns 0 if TIME ≤ $T_0$, else it returns Slope * (TIME – $T_0$).

Note: TIME is a function that returns the current time of simulation. See TIME.

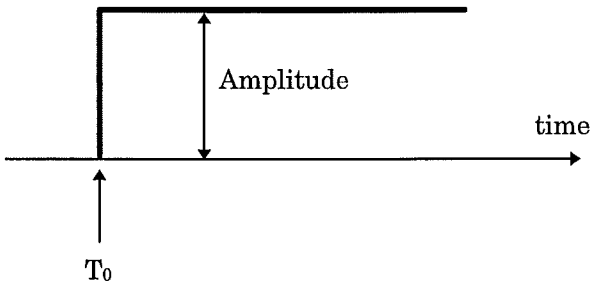



STEP - Step function

Syntax: STEP(Amplitude, $T_0$)

Result: Generates a step at time *To* with amplitude *Amplitude*. That is, it returns 0 if TIME ≤ $T_0$, and Amplitude otherwise.

Note: TIME is a function that returns the current time of simulation. See TIME.




TIME - Current time of simulation

Syntax: TIME

Result: Gives the current time of simulation.

TIMESTEP - Step length

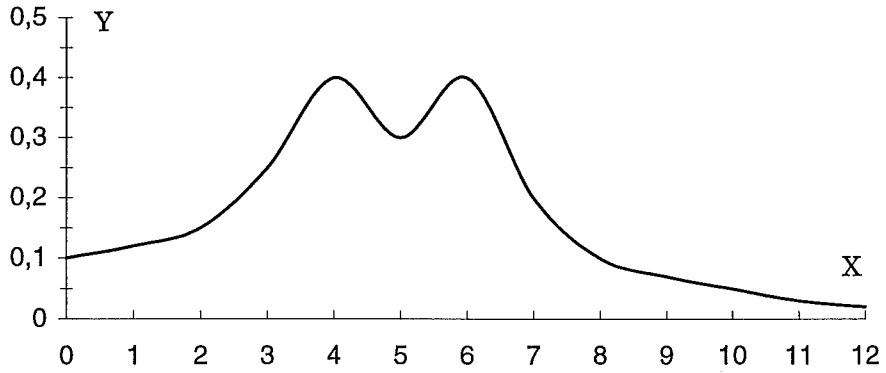
Syntax: TIMESTEP

Result: Gives the step length.

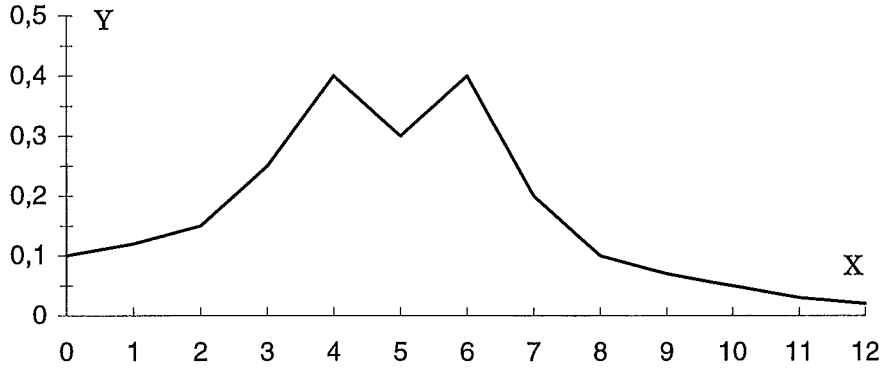
## Table functions

It is often necessary to express one variable in terms of its arbitrarily relationship to another variable. This relationship can be most easily expressed with a graph.

Consider for example the following relationship between X and Y:




In this case Y cannot be expressed as a function of X, i.e. Y = f(X), since we do not know the function f. Instead we approximate f with a suitable number of line segments. We will then have this relationship:




All table functions use equally spaced values of the independent variable, X, and the corresponding values of the dependent variable, Y, to define the relationship. The table below shows the idea. X is the input variable to the function and Y is the output variable.

| X-values | Y-values |
|---|---|
| $X_1$ | Y(1) |
| $X_1 + dx$ | Y(2) |
| ⋮ | ⋮ |
| $X_1 + (N - 1)dx$ | Y(N) |

GRAPH - Graph with linear interpolation and horizontal asymptotes

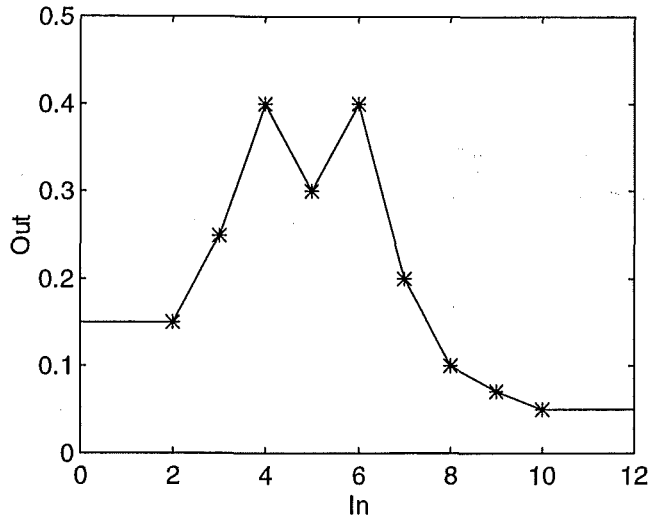
Syntax: GRAPH(In, $X_1$, dx, Y)

Result: GRAPH has horizontal asymptotes: If *In* < $X_1$ then Y(1) is returned, if *In* > $X_1$ + (N - 1)**dx* then *Y(N)* is returned.

If *In* is between two X-values then the returned value is calculated using linear interpolation.

Note: Y is a vector of length N. It is defined by [Y(1), Y(2), ..., Y(N)].

Example: GRAPH(In, 2, 1, [0.15, 0.25, 0.40, 0.30, 0.40, 0.20, 0.10, 0.07, 0.05]) gives the relationship below between the input value In and the returned output value Out:



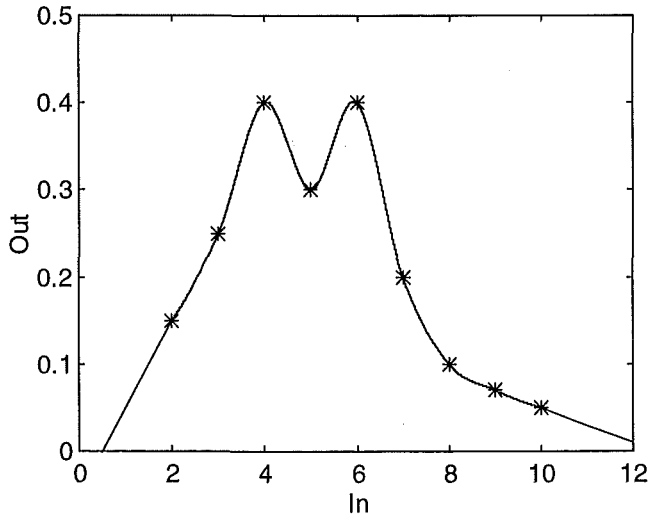

## GRAPHCURVE - Graph with polynomial interpolation and linear asymptotes

Syntax: GRAPHCURVE(In, $X_1$, dx, Y)

Result: GRAPHCURVE has linear asymptotes: If *In* < $X_1$ then a linear extrapolation using *Y(1)* och *Y(2)* is made, if *In* > $X_1$ + (N - 1)**dx* a linear extrapolation using *Y(N-1)* och *Y(N)* is made.

If *In* is between two X-values then the returned value is calculated using interpolation with a third-order polynomial.

Example: GRAPCURVE(In, 2, 1, [0.15, 0.25, 0.40, 0.30, 0.40, 0.20, 0.10, 0.07, 0.05]) gives the following relationship between the input value In and the returned output value Out:



GRAPHLINAS - Graph with linear interpolation and linear asymptotes

Syntax: GRAPHLINAS(In, $X_1$, dx, Y)

Result: GRAPHLINAS has linear asymptotes: If *In* < $X_1$ then a linear extrapolation using *Y(1)* och *Y(2)* is made, if *In* > $X_1$ + (N - 1)**dx* a linear extrapolation using *Y(N-1)* och *Y(N)* is made.

If *In* is between two X-values then the returned value is calculated using linear interpolation.

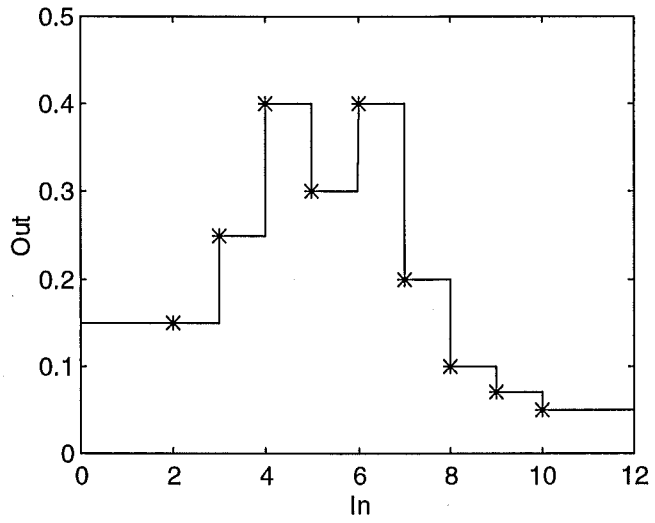
GRAPHSTEP - Horizontal graph with horizontal asymptotes

Syntax: GRAPHSTEP(In, $X_1$, dx, Y)

Result: GRAPHSTEP has horizontal asymptotes: If *In* < $X_1$ then Y(1) is returned, if *In* > $X_1$ + (N - 1)**dx* then *Y(N)* is returned.

If *In* is between two X-values then the returned value is the Y-value that corresponds to the lower X-value.

Example: GRAPHSTEP(In, 2, 1, [0.15, 0.25, 0.40, 0.30, 0.40, 0.20, 0.10, 0.07, 0.05]) gives the following relationship between the input value In and the returned output value Out:




## Delay functions

There are two types of time delays: one is associated with the time it takes to process physical material, and the other is assciated with time it takes to receive, evaluate and act upon information. These two types are called material delay and information delay, respectively.

Material delay is for example the time from that a product is ordered until it is delivered, or the time from that a person becomes infected until he becomes sick.

Information delay is for example the time under which one has to gather sales information before a decision of a changed production rate can be made.

---

DELAYINF - n-th order information delay

---

Syntax: DELAYINF(In, Delay, n, Init)

Result: Information delay. The input *In* is delayed using an *n*-th order exponential delay. The delay time is in average *Delay*.

When TIME < *Delay* then *Init* is returned. If *Init* is not specified *In* is returned.

Note: *In* must be linked to the delay variable in the Powersim diagram with a delayed link.

If *Init* is not specified then *In* must also be linked to the delay variable with an initialization link to show that *In* also is used to initiate.

Example: See DELAYPPL !

---

DELAYMTR - n-th order material delay

---

Syntax: DELAYMTR(In, Delay, n, Init)

Result: Material delay. The input *In* is delayed using an *n*-th order exponential delay. The delay time is in average *Delay*.

When TIME < *Delay* then *Init* is returned. If *Init* is not specified *In* is returned.

Note: See DELAYINF !

Example: See DELAYPPL !

---

DELAYPPL - Ideal material delay (pipeline)

---

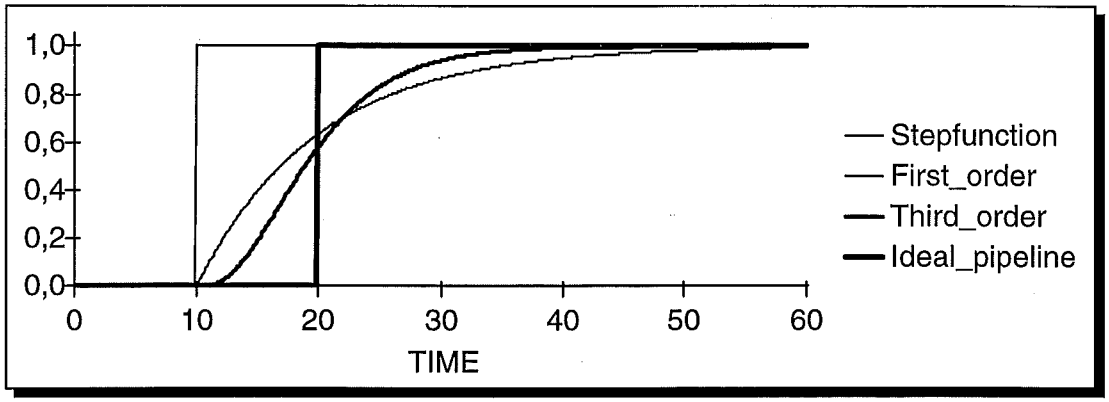
Syntax: DELAYPPL(In, Delay, Init)

Result: Material delay. Gives the value of the input *In* at *Delay* time units earlier in the simulation, i.e. *In* is delayed *Delay* time units.

When TIME < *Delay* then *Init* is returned. If *Init* is not specified *In* is returned.

Note: See DELAYINF !

Example: Below is the difference between delays of different orders shown when a unit step is the input *In*. Note that an ideal delay corresponds to an exponential delay of infinite order, i.e. $n \rightarrow \infty$.




## Rounding

CEIL - Round up

Syntax: CEIL(X)

Result: Returns the smallest integer that is greater than or equal to *X*.

Example: CEIL(2.3) = 3
CEIL(-4.3) = -4

FLOOR - Round down

Syntax: FLOOR(X)

Result: Returns the largest integer that is smaller than or equal to *X*.

Example: FLOOR(2.3) = 2
FLOOR(-4.3) = -5

ROUND - Round to nearest integer

Syntax: ROUND(X)

Result: *X* is rounded to the nearest integer.

Example: ROUND(2.3) = 2
ROUND(-4.3) = -4

## Logical functions

| IF - Arithmetic If | |
|---|---|
| Syntax: | IF(Condition, Expression1, Expression 2) |
| Result: | If the logical condition *Condition* is true (True) then *Expression1* is evaluated and returned, if it is false (False) then *Expression2* evaluated and returned. |
| Note: | The different kinds of logical conditions are described below. |
| Example: | IF( A < B, 5, FLOOR(6 * PI) ) returns 5 if A < B, otherwise 18. |

| <, >, <=, >=, =, <> - Logical conditions | |
|---|---|
| Syntax: | A < B, A > B, A <= B, A >= B, A = B, A <> B |
| Result: | Returns True if the condition is true, otherwise False. |
| Example: | A <> B will return True if $A \neq B$ and False if A = B. |

# 4 How to use Powersim in practice

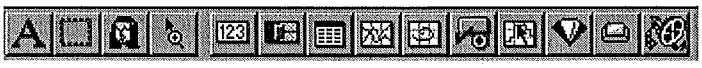
### Menus and toolbar

When implementing a model in Powersim the menus and the toolbar are used. The toolbar consists of two rows of buttons. The buttons on the first row are...

new, open, save, undo, cut, copy, paste, zoom out, zoom in, zoom to 100 %, zoom to fit window...

... straightens selected links, rotate name, move valve, play, run step, pause, stop, modify and autoreport.

The Powersim symbols are found to the left on the second row of the toolbar. They are:
Level, auxiliary, constant, flow-with-rate, flow, link and delayed link. Sinks and sources come automatically with the flows. To "draw" a symbol it is just to "fetch" it from the toolbar. The remaining buttons on this row are: Snapshot, eraser, pointer, chain, DDE object, archive object, network game...



...text, frame, picture, line, number, slider/bar, timetable, time graph, scatter graph, button, array graph, gauge, 3D button and multimedia object.

Everything that can be done by using the toolbar can also be done using the menus. It is easiest however to use the toolbar as much as possible. To find out more about a specific symbol use the On-line Help.

The Powersim symbols like constants, auxiliaries, initial values etc. are defined by double clicking on the corresponding symbol in the Powersim diagram. Then the dialog box ***Define variable*** appears. See figure 9.

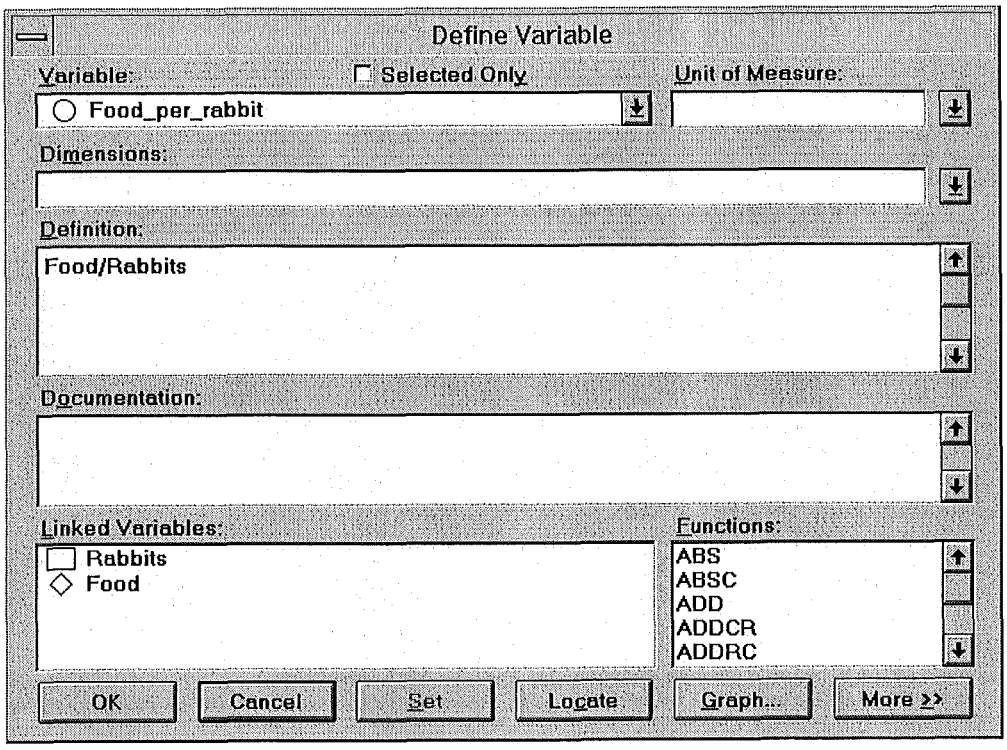


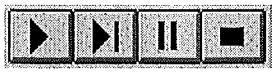

**Figure 9.** The Define Variable dialog box.

## Equations

Powersim automatically generates the equations that describe the model. These equations can be viewed by choosing **Equations** in the **View** menu. By choosing **Diagram** in the same menu the diagram reappears.

**Simulation**

To control the simulation there is a group of four buttons to the right on the first row in the toolbar. They signify RUN, RUN ONE STEP, PAUSE and STOP. See below.



The integration method and simulation length are chosen in **Simulation Setup** in the **Simulate** menu.

The available integration methods are
- Euler integration
- Runge-Kutta integration of 2nd-, 3rd- or 4th-order with fixed step length
- Runge-Kutta integration of 4th-order with variable step length

**Time graph, scatter graph, array graph, slider, gauge and table**

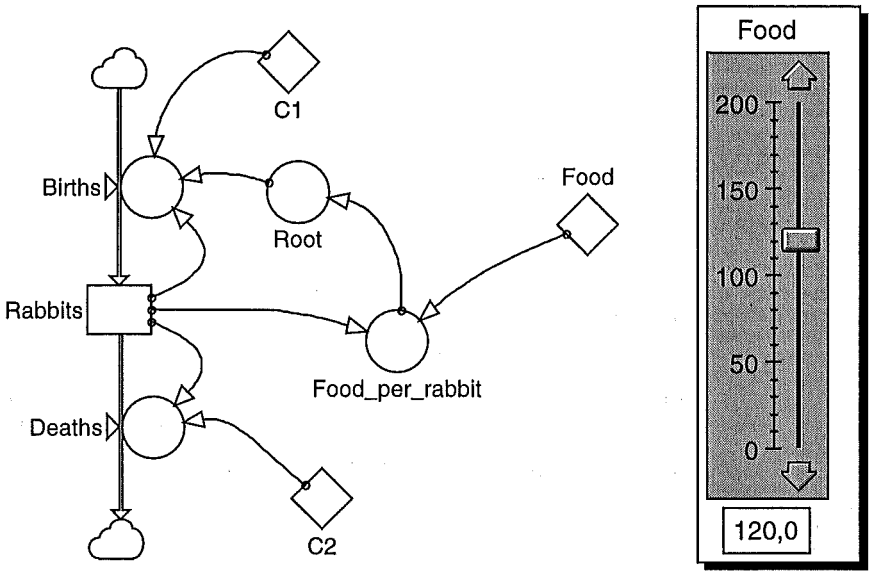
The result of a simulation can be visualized with a *time graph.* A graph can be placed anywhere. By double clicking on the graph a dialog box appears where you can choose the variables that you want to plot. You can also set the axes, change line types and colours etc. Two variables can be plotted against each other using a *scatter graph.*

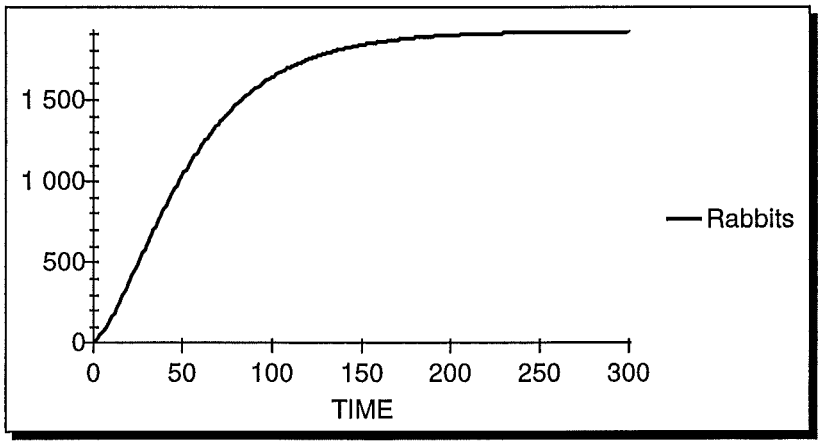
All constants can be controlled by a *slider* or *gauge*. By double clicking on the object a dialog box appears where you can choose the appearance of the slider, the range of the constant etc. To change the slider, i.e. change the value of a constant, *simulation-mode* must be entered. This is done by first pushing the PAUSE-button and then the RUN-button. This simulation is started by releasing the PAUSE-button.

If numerical values of the simulation results are needed a *table* can be used. By double clicking on the table the interval of the tabulated data can be chosen.

**Example**:

The example with the rabbit model presented in chapter 2 can have the solution presented below in figure 11, if using a time graph, a slider and a table. The slider in figure 11 is set so that the constant Food can be changed in the range 0 - 200 kg. The table is set so that the results are shown for every 5 months.




| TIME | Rabbits | Food_per_rabbit |
|---|---|---|
| 0 | 10,00 | 12,00 |
| 5 | 63,01 | 1,90 |
| 10 | 147,72 | 0,812 |
| 15 | 252,00 | 0,476 |
| 20 | 367,03 | 0,327 |
| 25 | 486,45 | 0,247 |
| 30 | 605,79 | 0,198 |
| 35 | 721,96 | 0,166 |
| 40 | 832,95 | 0,144 |
| 45 | 937,49 | 0,128 |
| 50 | 1 034,87 | 0,116 |
| 55 | 1 124,80 | 0,107 |
| 60 | 1 207,28 | 0,0994 |

**Figure 10.** The rabbit model presented with a time graph, a slider and a table.

## Arrays

If your model contains many identical or almost identical structures, it might be wise to use array variables instead of scalar variables.

Lets say we want to make a model of two competing animals, foxes and wolves, this would involve two similar structures - the main differences being values of parameters (constants) and levels (accumulators). Once a model of one animal is developed, more species may be added just by adding an animal dimension to every variable of the model.

Working with scalar variables a model could look like the one in Figure 11. (The competition factor is included in the constants.)

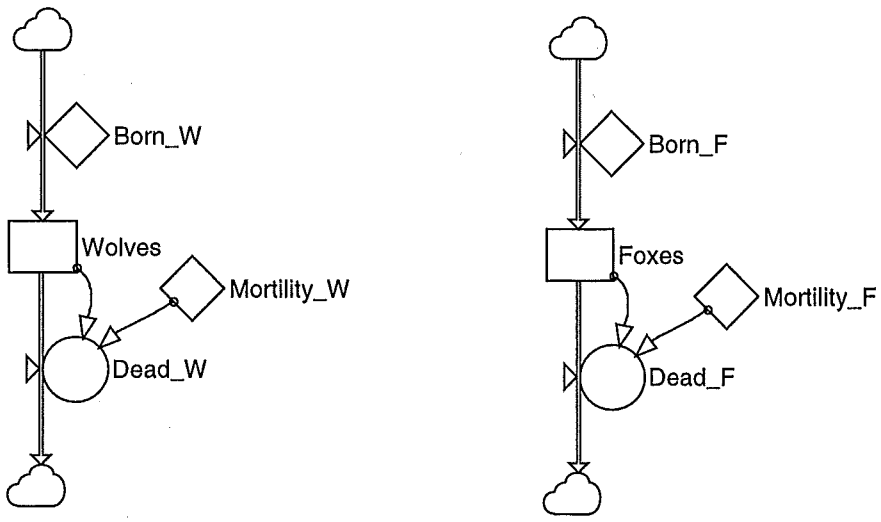



**Figure 11.** Two diagrams with scalar variables.

Instead of doing many similar structures we can define our variables to contain an array of "cells" where each element is stored in its own "cell" (see Figure 12). The double lined symbols tell us that we are dealing with arrays.

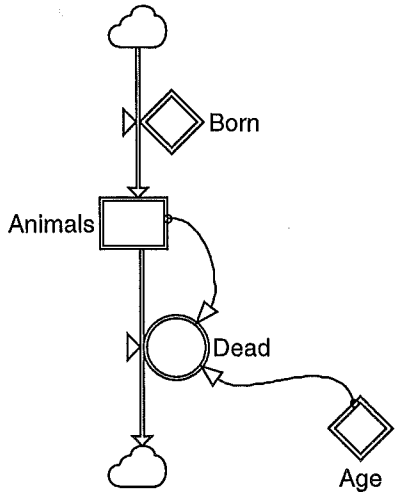



**Figure 12.** The same diagram as above but with array variables.

Arrays are defined by specifying one or more *dimensions*. Each dimension is defined as a *range* of values. It is possible to define enumeration or numeric ranges. An enumeration range might for example be child, young, adult, old.

When building an array model you have to do the following:
First of all check that the Simplified User Interfaced in the View menu is disabled.

1. Build the scalar version of your Powersim diagram.
2. Open the Edit **Define Range** dialog box.
3. Press the *Add* button and type in the name of the new range (Animals).
4. You can choose a numeric subrange, enumeration subrange or enumeration. Select *Enumeration* in the **Range Type** box.
5. Type the name of your first variable (Foxes) in the **Element Name** text box and hit return.
6. Type the next one (Wolves) in the **Element Name** text box and hit return, and so on,
7. Close the dialogue box by using the OK button.
8. Open the **Define Variable** dialog box by double-click on the different variables (e.g. Born)
9. Type the name of the new range (Animals) in the **Dimensions box**.
10. Define your variable in the **Definition field**, e.g. [2, 4] telling Powersim that two foxes and 4 wolves are born every year.

    For every symbol you have to define the variable by;
    A) typing the *New range* (Animals) in the **Dimension box** and,
    B) typing either numbers [1, 4, ...] (for starting values of state variables and constants) or functions (when defining flow and auxiliary variables, e.g. Age * Animals) in the **Definition field**.

11. Define all your variables like above. When your diagram is ready you can create a Time Graph object and drag the Animals variable into it and run a simulation.

*Example*
Lets say we have an array of 5 elements and the initial valus are [1,2,3,2,1]. Now we want to transport the values to the right. Thus it will look like the "squarewave" is moving towards the right. The values will go down to 0 when the wave has passed.

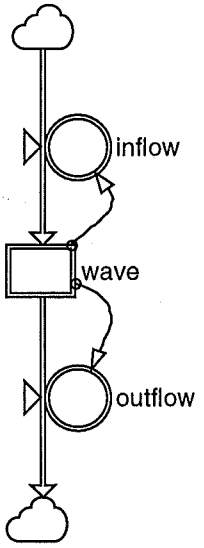
First of all define a range. Go to Edit and Define Range:
rum = (1..5) (a numeric subrange)

Then place a level variable, called wave, and an in- and outflow from the wave level.
Double click on the level and fill in
1) dimension: rum
2) startvalue: [1,2,3,2,1].
Double click on the inflow and fill in
1) dimension: i=rum
2) definition: 0 WHEN i=wave(1) BUT rum (i-1) OTHERWISE
Double click on the outflow and fill in
1) dimension: i=rum

2) definition: rum(i)

The flows have to be devided with the TIMESTEP if TIMESTEP ≠ 1, since Powersim takes the flow times the TIMESTEP for each iteration.

**Powersim model**




```
Dim  wave  = (length)
Init  wave  = [1,2,3,2,1]
Flow  wave  = -dt*outflow
              +dt*inflow
Dim  inflow = (i=length)
Aux  inflow = 0 WHEN i=length(1) BUT wave(i-1) /TIMESTEP
              OTHERWISE
Dim  outflow = (i=length)
Aux  outflow = wave(i)/TIMESTEP
```

**Multiple runs**
A series of simulation runs can be set up in order to perform an optimization or sensitivity analysis. If you for example want to compare simulation results with different integration methods you can do like this:

1. Create a **Time Graph** object and open its *Definition dialog box.*
   1.1 Select the variable you want to study in the *Parameters list box*
   1.2 Switch on *Generations Simulations Starts New Generation*
   1.3 Switch on *Display Add New Generations to Display List*
2. From the Simulate menu choose *Integration method* and *Time Step*
3. Simulate
4. Repeat step 2 and 3 as many times as necessary

From the **Run Setup** dialog box it is possible to specify that a series of simulation will be performed in a row. This feature is used mainly for testing the behavior of the model with varying parameters of inputs. Say for example that you want to vary the value of a constant from 1 to 20 in steps of 2 and study the results from 10 simulations. This is accomplished by expressing the constant in terms of the function RUN (constant = 2*RUN), where RUN is the simulation number.

**Co-models**
Powersim provides means of synchronizing individual models during simulation. This is done by hooking co-models up to a main model. A co-model is an independent model that may be simulated in the normal way. In the **Simulation Setup** dialog box other models may be connected to the current model, which becomes the main model. When the main model starts

simulating the co-models are automatically simulated as well. The simulation time of the co-models are mapped to the time interval specified for the main model. This ensures that the main model and the co-models will start and stop simulating at the same time.

A possible reason for splitting a model into a main model and a set of co-models is the need of different integration methods for different parts of the model. One part might require a high integration order and a small time step. If the model is large, it may cause problems if the time-consumption integration method is used for the whole model. In some cases this problem can be solved by splitting the big model into one model that uses a high order integration method with a small time step, and one model that uses a less accurate but faster integration method and a larger time step.

To use different integration methods for different parts of a model you do like this:
1. Split your model into two parts, say MAIN.SIM and CO.SIM
2. Open the MAIN model and make its window active (on top)
3. Open the **Simulation Setup** dialog box
4. Add CO.SIM as a co-model of MAIN by using the *Add* button
5. Select MAIN in the *Parallel Simulation* list box and define integration method (e.g., Euler with a time step of 1)
6. Select CO in the *Parallel Simulation* list box and define integration method (e.g., RK4-Var Step with a time step of 0.025)
7. Close the Simulation Setup dialog box and run the simulation

## Data transfer

### ... between models

Transferring data between different Powersim models is done by adding instances of the transfer object chain to the model. The chain object will connect a variable in the model to a variable of another model. Both source and destination variable may be chosen from either the main model or one of the co-models. To establish a connection between a main- and a co-model.
1. Add one or more co-models using the Simulation Setup dialog box.
2. Create a **Chain** object and put it in the diagram.
3. Open the **Define Chain** dialog box, choose the *source* and *target* variable, and hit OK.

When simulating the main model, the destination variable will now have its value transferred from the source variable.

### ... to applications

Data can be transferred to and from Powersim using the clipboard and Dynamic Data Exchange (DDE). If you want to copy the data to other applications, e.g., to spreadsheets or word processors use *Copy* which copies the time series data selected from the Displayed Series list. Powersim diagram can also be included in other applications as embedded or linked objects (OLE).

The **DDE object** is used for transfer of data between applications. You can use the DDE object for several purposes, including:

- Retrieval of parameter values from external applications (e.g., spreadsheets)
- Monitoring external events
- Controlling external devices
- Visualizing variable values through external applications
- Exporting data to external applications for analysis or storage

To define the DDE object, double-click on it. Now you will see the Define Dynamic Data Exchange dialog box.

1. Chose model
2. Chose the variables you want to export
3. In the *Transfer box* you chose if you want to export or import data.
4. In the *Values group* you can define if you want to transfer a single value or a time series.
5. Use the *Service box* to enter the name of the application with which you are exchanging (for example Excel).
6. Use the *Topic text box* to define the name of the application file (e.g. test.xls).
7. Use the *Item text box* to specify the cells where the data is to be placed. The convention used is to refer to the cell's row and column location, separated with a colon. r5c1:r20c2 would be row 5 and column 1 to row 20 and column 2, that is A1 to B20 in Excel..

**...to file**

The **Archive object** connects simulation to an external data file. The Archive object is used to transfer parameter values to or from external files. Both single values and time series can be transferred. Examples of uses:

- Initializing variables with data stored in file
- Loading data for variables from external time series during simulation
- Saving parameters to file
- Saving time series to file

The Define Archive object dialog box works similar as the Define DDE dialog box. Simulation results may be exported in order to make statistical analysis and comparisons with collected data.

**Snapshot**

A variable symbol with an extra set of corners is called a snapshot. It is a picture of or an alias for an original variable. Snapshots are useful for linking variables located in different parts of a model.