Module 2 :: INSEL programming concepts

The INSEL idea is based on a modular, block-oriented concept which adapts structured programming – a programming method which restricts algorithms to three basic programming structures, i. e. (i) sequence structures, (ii) if-then-else structures, and (iii) loop structures – to block diagrams. In computing science it has been shown, that all numerical problems can be solved with these three basic structures. Therefore, INSEL is a general-purpose programming language, which can – in principle – be adapted to any numerical task.

2.1 INSEL block groups

Although all INSEL blocks appear as named rectangles with inputs, outputs and parameters, each block belongs to a certain block group.

The following six block groups exist in INSEL:

- Constant blocks or short C-blocks
- Timer blocks or T-blocks
- Standard blocks or S-blocks
- Loop blocks or L-blocks
- Delay blocks or D-blocks
- If blocks or I-blocks

S-blocks

As the name indicates already, the group of S-blocks is the least specific. In Module 1 we have used the SIN block and the PLOT block, for instance. There is nothing special about them. They are typical S-blocks. When the SIN block gets an input value α, it calculates the corresponding sine value and connects the result with the block's output – finished. When the PLOT block gets a data point with coordinates x and y as inputs, it plots the data point – finished. But who delivers the inputs and how often – and who decides when the simulation run is through?

T-blocks

In the simple DO–SCREEN example of Module 1 we have specified by the parameters (initial value 1, final value 10, increment 1) the DO block "fires" 10 times: a 1 in the first step, a 2 in the second step, a 3 in the third step, and so on until the block outputs a 10 in the 10^{th} step. Then there is nothing left to be fired – hence the DO block sends a signal to the inselEngine to end the run. Blocks having the ability to control a simulation model are called timers, or Timer blocks or just T-blocks.

It is not compulsory to include a Timer block in an INSEL application. The following example does not use a Timer block, for instance.

This simple application uses three blocks:

- The CONST block, which just delivers a constant output as specified by a parameter, 45 (which we humans interpret as 45°) in this case.
- The SIN block, which calculates the sine of its input.
- The SCREEN block, which can be used to display alphanumerical information on the computer's screen.

Test it. You find the CONST and SIN block in the Mathematics category of the palette under Constants – Any constant, and under Trigonometric functions – Sine, respectively. The SCREEN block can be found in the Inputs and outputs category as Screen output.

When you run the model you will see the output 0.70710677 from the SCREEN block, which means that you have calculated $\sin(45°) = 0.70710677$.

Sorting

INSEL has executed every block one time: The CONST block which defines the output 45, the SIN block which calculates $\sin(45°)$ and the SCREEN block which displays the result – ready. Please observe that INSEL calls the blocks exactly in the order CONST, SIN, SCREEN, no matter in which order you have constructed the block diagram.

This means that there must be some mechanism in INSEL which converts a block diagram description into a calculation list – this mechanism is called sorting algorithm and is an integral part of the inselEngine. You will learn more about the inselEngine in due course.

C-blocks

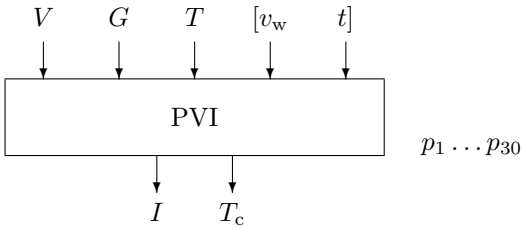
No matter whether there is a T-block in a model or not the CONST block always needs to be executed only once and never again. Blocks with this property belong to the group of C-blocks.

We can conclude that we have seen examples of a C-block (the CONST block), S-blocks (the SIN and SCREEN block), and a T-block (the DO block). This Module deals only with these three block types. Loop, Delay, If, and Macro blocks will be handled later.

2.2 Basic photovoltaics

So far, we have used only quite primitive blocks. One of the nice aspects of INSEL however, is that basically all blocks look alike and can be treated more or less in the same way, regardless of whether they are primitive like the CONST block or more complex like the PVI block, which we had used already in the previous Module.

The PVI block is located in the category Electricity under Photovoltaics – Photovoltaic current (c-Si). This is the design of the block:



As indicated by the bitmap of the PVI block in the left margin, this block simulates the behavior of solar cells – PV is short for photovoltaics, i. e. the direct conversion of electromagnetic radiation into electricity. The sketch of the PVI block shows, that this block requires (up to) five inputs and – don't get shocked – 30 parameters.

With this information the block calculates two outputs, the PV current I in ampere (this I gives the block its name) and the cell temperature T_c in degrees Celsius. PVI is also a Standard block.

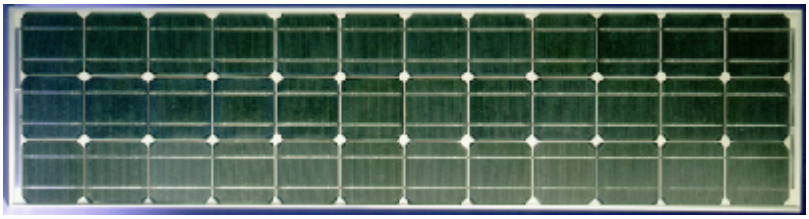
In order to use the block it is necessary to connect at least three inputs: the voltage V of the device in volt, the global radiation G in W/m^2. T stands for a temperature in degrees Celsius. This input has a specific role and will be discussed later.

Two-diode model

Concerning the parameters: The electric properties of the solar cell are modeled by a rather detailed physical model, well-known as the two-diode model. For the calculation of the thermal properties of the device an energy-balance differential equation is used. In addition, the block can be used for any particular electrical connection of cells and modules in series and in parallel. All in all this gives 30 parameters.

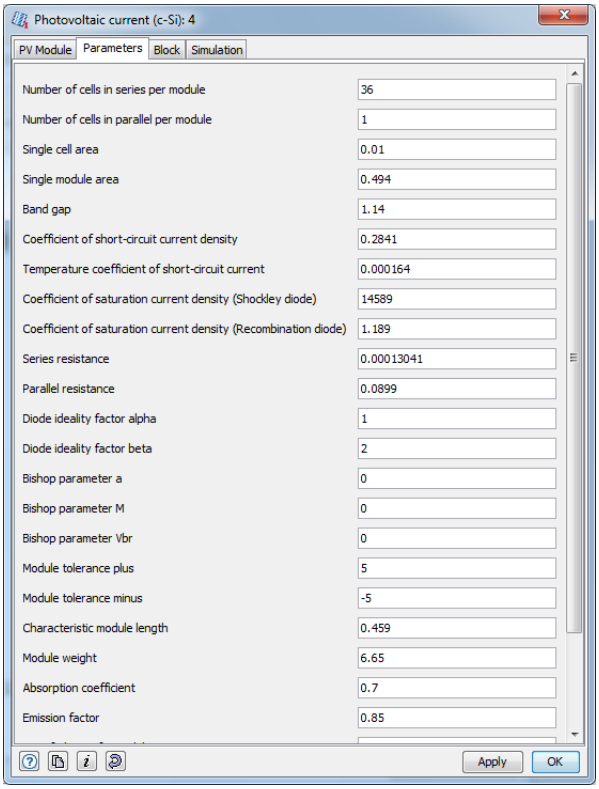
We will not go into the details here. More information about PV modeling in INSEL can be found in Module 7.

INSEL contains more than five-thousand parameter sets for practically all modules that are available on the market or have ever been produced. The photo shows one of these modules – the Siemens module SM 55.

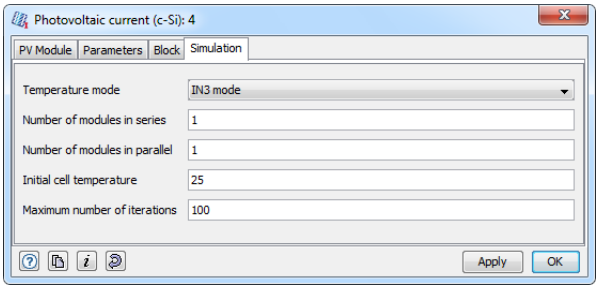


For historic reasons, the default parameter set used by the PVI block simulates exactly this module.

The “physical” parameters can be seen after a double-click on the PVI block on the parameters pane:



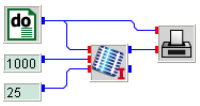
The “variable” parameters can be accessed via the Simulation pane:



As we have seen, the PVI block calculates the PV current I as a function of the voltage V. Analogue, there is a block called PVV (Photovoltaic voltage (c-Si)) which calculates the PV voltage V as a function of the current I. It always depends on the actual problem, which one is better to use.

I-V curves

We have already used the PVI block in the previous Module for a plot of the voltage-current characteristics, the I-V curve under standard test conditions STC (defined as global radiation equal to 1000 W/m^2 at a spectral distribution of AM 1.5 and a module temperature of 25 °C). We repeat the block diagram:



The DO block is used to vary the voltage in a range between 0 and 25 volt in steps of 10 millivolt. Two CONST blocks provide values for the global radiation and the module temperature. The PLOT block is used to display the I-V curve.

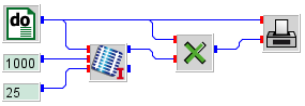
The parameter *Temperature mode* is set to IN3 mode, by default, which means that the module temperature is given by input number 3 – which comes from a CONST block with value 25. The other temperature modes will be discussed later.

Exercise

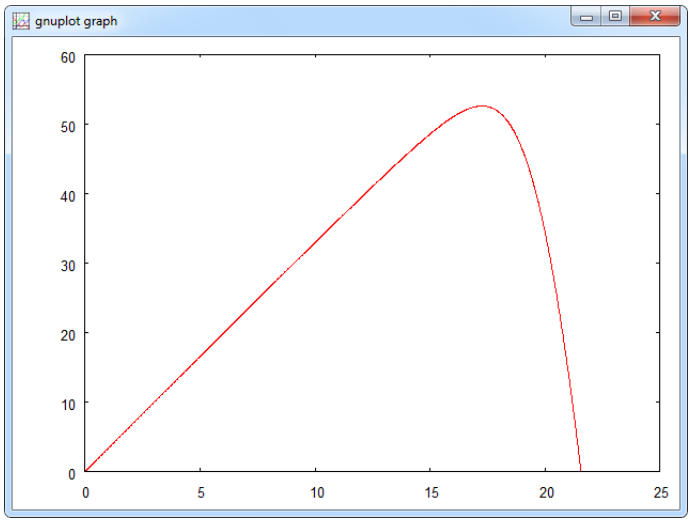
Now, please reconstruct the block diagram from scratch and run it until you see the I-V curve displayed by the PLOT block.

DC power

It is now easy to calculate and plot the DC (direct current) power output $P_{\mathrm{DC}} = I \cdot V$ of the module as a function of the operation voltage. All we need to do is to multiply the first output of the PVI block (the current I) with the output of the DO block (the voltage V). This can be done with the S-block MUL which we know already from the previous Module.



This block diagram solves the problem and shows the DC power as a function of the voltage.



We see from the graph that the output power depends very much on the operation voltage with a maximum of about 53 W_p (Watt peak) close to 17 volt. This point (V_m, I_m) is called the maximum-power point in photovoltaics and defines the peak power or nominal power of the module under standard test conditions. Under real operating conditions the maximum-power point varies, since it depends on global radiation and module temperature.

In most real PV generators there will be a device called maximum-power-point tracker which will always operate the generator close to this point. In a numerical simulation this operating point must be found by an iteration process. The INSEL block which performs this iteration is called MPP. This Loop block will be handled in Module 5.

There is another INSEL concept that can be learnt from the PVI block.

2.3 The INSEL concept of time

In a real-world PV generator the module temperature will depend on the weather conditions and will be a function of time. When you look at the temperature modes of the PVI block, you find the DEQ mode (differential equation). In this mode the module temperature is calculated as a function of voltage V, global radiation G, ambient temperature T_a, wind speed v_w, and time t.

So far, the PVI block we used had only three inputs. Hence, two more inputs for the wind speed v_w and time t are required. Remember, we can define the number

of block inputs through the blocks pane.

Time in seconds

In the classical simulation environments like CSMP and SPICE, for example, and even in most modern ones like MATLAB and Simulink time plays an extra-ordinary role. In INSEL time is just a variable among others. It is always an explicit block input, which has a time-dependent behavior. As a general rule, time in INSEL always runs in seconds.

Variable time steps

In temperature mode DEQ the PVI block must be supplied with a time input in seconds. If, for example, you want to run a PVI block in time steps of one hour, you must deliver values like 0, 3600, 7200, and so on to the PVI block.

Since the PVI block is able to remember the value of the time input of the latest call, the block itself can calculate the actual time difference between the previous call and the actual one, i. e. the time step. A consequence of this concept is that the time steps of a simulation run in INSEL do not at all need to be constant.

The PVI block can deal with any time step, no matter whether the time step is in the range of seconds or hours.

INSEL Lab

It's time for a concrete example. Let us observe how an SM 55 PV module heats up with time.

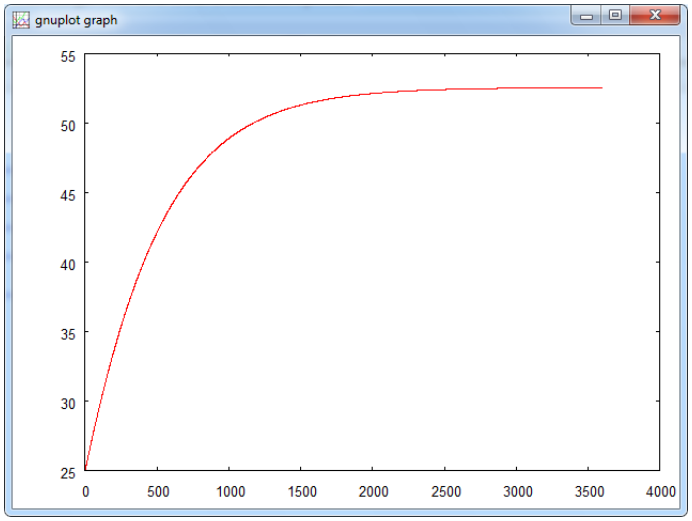
You will gain the highest benefit from INSEL when you do not think of writing simulation applications, but perform "close-to-real experiments" in a laboratory.

Let us place a Siemens SM 55 module in a laboratory environment, and wait until it is in equilibrium with ambient conditions, assumed to be $T_a = 25\,^{\circ}\mathrm{C}$, no air movement, i. e. $v_w = 0$ m/s and completely dark, i. e. $G = 0$ W/m^2. This is equivalent to setting the initial value for the cell temperature parameter of the PVI block to 25 degrees Celsius.

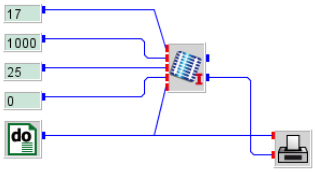
We then switch on a light source which illuminates the module with 1000 W/m^2, for example. In a real experiment we could use a PT-100 for the module temperature measurement, in INSEL the PVI block with the parameters of the Siemens SM 55 module provides the cell temperature T_c as a second output.

After about one hour we would expect equilibrium conditions for our experiment. Hence, let us run the simulation for one hour. Would you like to solve the problem yourself or just look at the solution?

Here is our solution:



The corresponding block diagram looks like this und moechte noch verschoenert werden:



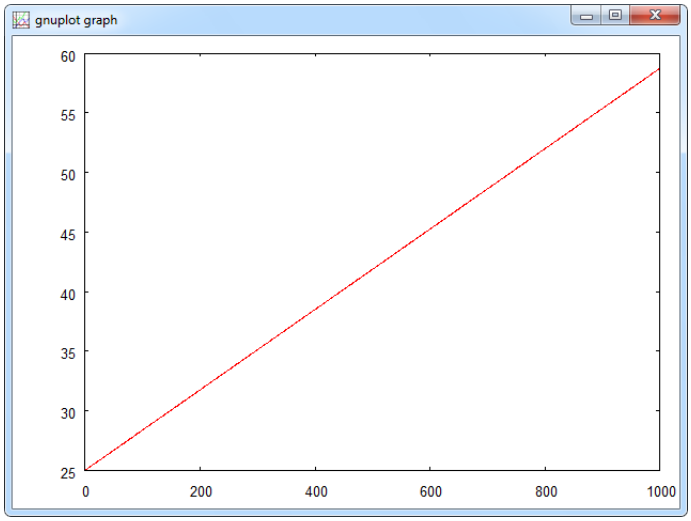
NOCT temperature

There is one last uncovered temperature mode of the PVI block, the NOCT mode. NOCT is short for nominal operating cell temperature. It is defined as the equilibrium module temperature under a global radiation $G_{\mathrm{NOCT}} = 800\,\mathrm{W/m^2}$, ambient temperature of 20 degrees Celsius and a wind speed of 1 m/s.

In NOCT mode the PVI block makes the linear interpolation

$$T_{\mathrm{c}} - T_{\mathrm{a}} = (T_{\mathrm{NOCT}} - 20\,^{\circ}\mathrm{C})\,\frac{G}{G_{\mathrm{NOCT}}}$$

You should quickly check the module temperature as a function of global radiation from 0 to 1000 W/m^2. For the voltage you can use 17 volts – we have seen before that the voltage near the maximum power point of the module is of that order.

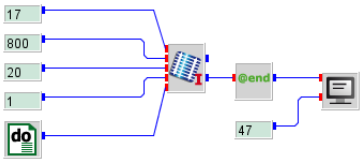


Exercise Adapt the DEQ mode example to NOCT conditions and compare the equilibrium temperature with the NOCT value.

Hint If you use the ATEND block with input T_c and connect its output to a SCREEN block, the SCREEN block displays only the last calculated temperature value. You find the ATEND block under the Mathematics – Logics category as At end. The ATEND block is already a first example of an I-block which will be discussed in more detail in Module 11.4.

Solution The result is 38.74 degrees Celsius compared to an NOCT of 47 degrees – frustrating or wrong?

This is the corresponding block diagram welches auch noch verschoenert werden moechte:

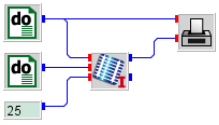


2.4 Nested Timer blocks

Timer block can be nested. This means that two or more T-blocks can be connected in series but not in parallel.

It's best to explain this with a concrete example: Assume that we want to use the PVI block to display not only one I-V characteristic but a set with the global

radiation as curve parameter. In the first place, one would simply replace the CONST block for the radiation by another DO block, and set the parameters of the new DO block to 200, 1000, with an increment of 200 W/m^2, for instance.



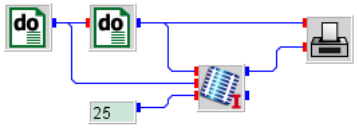
When you run this block diagram, INSEL will generate an error message which says "Too many timer blocks specified". Why?

The two DO blocks are not connected in series but in parallel. It is not clear how INSEL should handle the model. Shall INSEL first fix the voltage value to zero and then run through all radiation data, return to the voltage block, increment the voltage to 0.01 volt, run through all radiation data again, and so on? Or shall INSEL first fix the radiation value to 200 W/m^2 and then run through all voltage values, return to the radiation block, increment the radiation to 400 W/m^2, run through all voltage values again, and so on?

From a curve plotting point of view, the second option is clearly better. But how can we express that we prefer the second option?

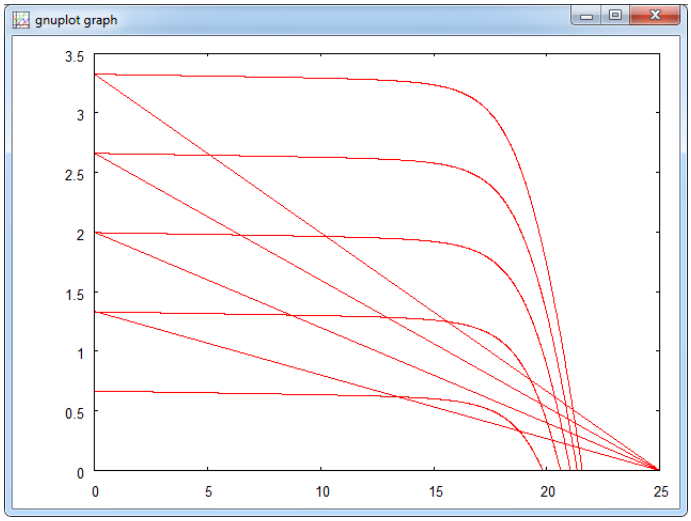
Key concept

One of the key concepts in INSEL is that blocks cannot be executed before all block inputs are "known," i. e. have values. So, if we add a new input port to the DO block which varies the voltage and connect it to the output of the DO block which varies the radiation, then the block for the voltage variation depends on the radiation block – they will be connected in series, nested!



Hence, the input of the DO block serves the purpose of arranging the two DO blocks in a fixed order.

When you run the model you will see some scrambled lines like this:



What happened? We have five blocks in the model:

- A CONST block which defines the module temperature
- A DO block which varies the global radiation
- A DO block which varies the voltage
- A PVI block which calculates the PV current
- A PLOT block which plots the data points

Calculation list

How does INSEL execute the blocks? The order in which the blocks in a model are executed is called calculation list in INSEL It can be displayed via the *Simulation – Show calculation list* menu.

```
Number    Block       Group    Jump
-----------------------------------
     5    CONST           C       1
     1    DO              T       1
     2    DO              T      -1
     3    PVI             S       1
     4    PLOT            S      -2
-----------------------------------
```

We see that at first the constant temperature value is set, then the first DO block with user block number 1 sets the radiation to 200 W/m^2, then the second DO block number 2 outputs a voltage of 0 V, then the PVI block calculates the PV current I, then the PLOT block plots the first data point ($x = 0, y = I_{sc}$), the short-circuit current. And then?

The last column in the calculation list is the so-called jump parameter. The value for the PLOT block is -2, i. e. INSEL returns control to the lower DO block number 2 in the calculation list. The DO block 2 increments the voltage to 0.01 V, the PVI block calculates the corresponding current, and the PLOT block plots the second data point, while drawing a linear interpolation line between the first and the second point.

This process continues, until the PLOT block gets the last data point from DO block number 2, which is equal to 25 V, and plots it – again with a short linear interpolation line between the last value ($x = 24.99, y = 0$) and the actual point. And then?

INSEL again returns control to DO block 2. But this block has nothing left to do. So, it gives control to the next "upper" T-block, which is DO block number 1. This block increments the radiation to 400 W/m^2, and DO block 2 gets control again.

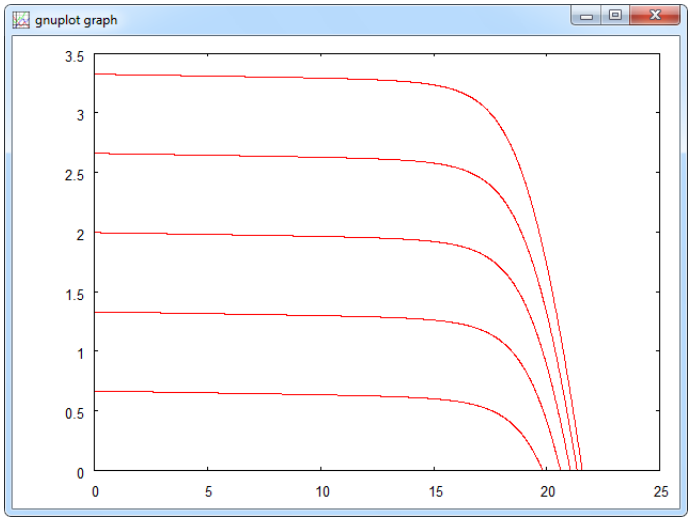
The DO block performs a reset since its input has a changed value and outputs its initial value again, PVI then calculates the short-circuit current $I_{sc}(400\ \mathrm{W/m^2})$, and the PLOT block gets the next data point ($x = 0, y = I_{sc}(400\ \mathrm{W/m^2})$) and operates as always: draws a linear interpolation line between the last point and the actual point – et voilà!

Parametric plot

The PLOT block had no chance to recognize that the radiation has changed and that we wanted to see a new line, i. e. simulate a pen-up pen-down operation.

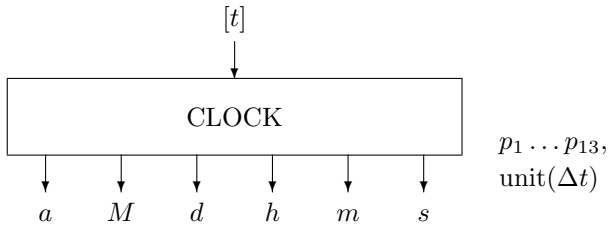
A way out is to "inform" the PLOT block about the curve parameter via the output of the DO block which varies the radiation. This means that besides the x- and y-coordinate the PLOT block requires a third input. The name of the block with such an extra input is PLOTP. Both, the PLOT and the PLOTP block can be found in the *Inputs and outputs* category of the palette as types *Gnuplot graph* and *Gnuplot graph (parametric)*, respectively.

So, replace the standard PLOT block by the parametric PLOTP block. The first input is the curve parameter – the output of DO block number 1 – the second input is the voltage – the output of DO block number 2, the third input the corresponding current. Now the result looks as it should look like.

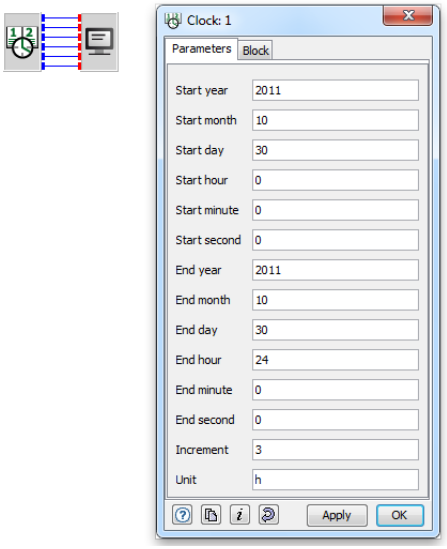


2.5 The Timer blocks CLOCK and FDIST

So far, the only timer block we have used is the DO block. A second example for an INSEL T-block is a block called CLOCK, found in the Time category as Clock. It behaves very much like the one you are probably wearing around your wrist: It runs through time in hours, minutes, and seconds, every day, month and year. The main difference is that a wrist-watch shows the time in which we humans are stuck. A simulation of a clock is much more flexible. We can let it run from any starting point to an end in any time step we like.



Start year, month, day, hour, minute, and second, end year, month, day, hour, minute, and second is the correct order of the parameters, followed by an increment Δt and a string for the unit of Δt. The unit must be one of these: a (for years), M (for months), d (for days), h (for hours), m (for minutes), or s (for seconds). If for example, we let the CLOCK run for one day from midnight to midnight in steps of 3 hours, for example, this object does the job:



We have added a SCREEN block so that we can observe what the CLOCK block does exactly.

```
Compiling clock.vseit ...
No errors or warnings
Running insel 8.1 ...
 2011.   10.   30.    0.    0.    0.
 2011.   10.   30.    3.    0.    0.
 2011.   10.   30.    6.    0.    0.
 2011.   10.   30.    9.    0.    0.
 2011.   10.   30.   12.    0.    0.
 2011.   10.   30.   15.    0.    0.
 2011.   10.   30.   18.    0.    0.
 2011.   10.   30.   21.    0.    0.
Normal end of run
```

Nothing spectacular happens. But you should take note of some details.

- We let the clock run exactly until 24:00:00 of 30 October 2011. This is absolutely equivalent to running the clock until exactly 00:00:00 of 31 October 2011.
- The CLOCK block runs in logical time steps and we are going to use it for exactly that purpose in most cases, for time step simulations. Logically we have defined a constant time step of three hours. The CLOCK block outputs the time information only once per time step and the time which is on output then is always the “left end” of the time interval, i. e. our first time interval is between 00:00:00 and 03:00:00 o’clock, but the CLOCK shows 00:00:00 “all the time.”

As a consequence, the last output of the CLOCK is at 21:00:00 for another three hours. At midnight, the clock stops. When you count the lines that the SCREEN block writes you find eight lines times three hours gives 24 hours – exactly what we wanted. Mathematically spoken, the CLOCK block runs through the interval with the left end closed, the right end open, i.e. [00:00:00,00:00:00).

Star format

- We have used the Format string (6F6.0) in the SCREEN block's parameter. This is a Format string in Fortran language standard. Fortran formats will be discussed in more detail in the next Module "Reading and writing data files."

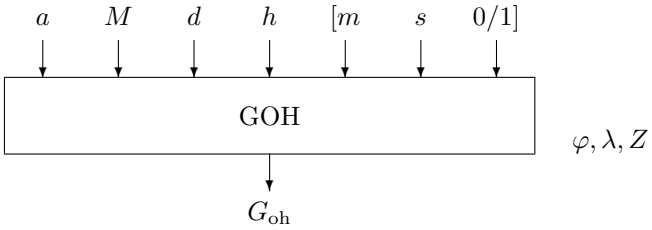
  For the time being, you should note that there is a so-called star format, which you can use easily by just typing in an asterisk * in the parameter field. Please try it, the star format is really useful for output when you do not know the order of magnitude of your results in advance.

You may ask "What are applications of the CLOCK block?" Here comes one which opens a huge field of applications of the block: Solar energy applications. In INSEL almost all of them make use of the Gregorian calendar.

2.6 Solar radiation

Radiation outside atmosphere

A simple example for a solar energy application is the block GOH which calculates the extraterrestrial radiation – that is the solar radiation in Space outside the Earth's atmosphere.



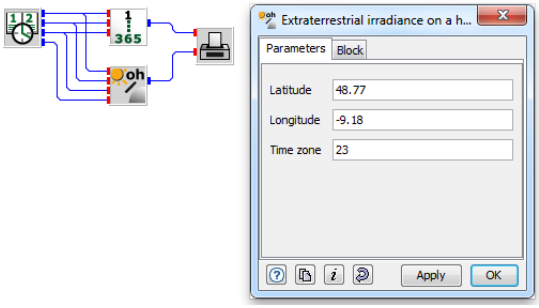
Inputs to the block are basically the outputs of the CLOCK block – at least the first four inputs are necessary. Required parameters are latitude φ, longitude λ, and time zone Z of the observer's location.

Latitude, longitue, time zone

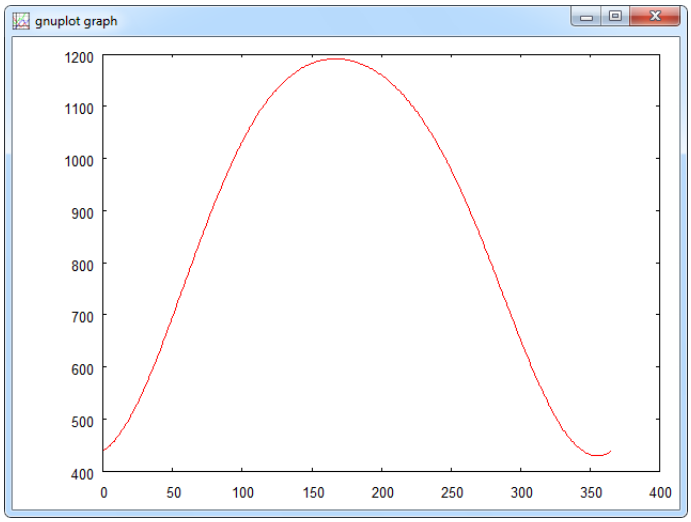
The latitude of an observer is defined from the equator towards the poles, northern hemisphere positive, southern hemisphere negative – Stuttgart in Germany has a latitude of about 48.77° north, for example. The longitude is defined as west of Greenwich, a cosy suburb of London in the U.K. We have to go almost all the way around the globe to reach Stuttgart at its longitude of 350.82°, but in INSEL we can also use −9.18° as longitude value for Stuttgart.

Time zones are also defined with respect to Greenwich defined as time zone zero – known as Greenwich Mean Time GMT – with approximately 15 degrees per time zone. The time on our German clocks shows Central European Time CET which corresponds to time zone 23. During summer we use daylight-saving time, which means we bring the time on our watches one hour ahead in spring, and put it back in fall. The last input of the GOH block is 0 by default (i. e. does not consider daylight-saving time). If you connect a one with this input GOH interprets the given time as daylight-saving time.

Let us calculate the annual course of the extraterrestrial radiation on a horizontal surface at noon for our home location – in our case this is Stuttgart with the given geographical parameters.



You find the DOY block (Day of the year) in the Time category, block GOH (Extraterrestrial irradiance on a horizontal surface) in category Meteorology – Solar radiation. The result is shown in the next graph.



You should grasp two points from this example:

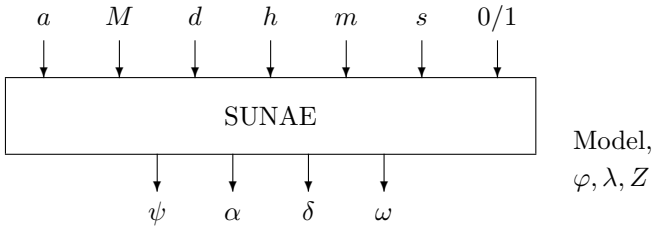
- A technical point: The CLOCK block is nice in time handling, it allows us to think of time like we are used to it. But for numerics it is rather bad. When we plot time series, we need a continuous signal, not something strange like the Gregorian calendar with all its exceptions, like leap years etc.

  INSEL offers several routines (blocks, of course) that handle this aspect. In the above example we have used the day of the year block DOY which converts a Gregorian calendar date to a continuous signal. Similar blocks are the hour of the year block HOY, and minute of the year block MOY, for example – all found in the Time category.

- A point of general interest: From the extraterrestrial radiation plot you can observe that in our place (Germany) the extraterrestrial radiation at the beginning of the year is much lower than in the middle of the year – there is a factor three between the values. The reason – which corresponds with our every-day-life experience – is due to the fact that in summer the Sun is much "higher" as compared to the winter case.

Solar position

The exact position of the Sun at any time can be calculated with the Standard block SUNAE (Meteorology – Geometry – Position of the Sun).



Horizontal system

The meaning of SUN in the block name is self-explaining, A stands for azimuth, in INSEL denoted by the Greek letter ψ, E stands for elevation, in INSEL denoted by α. Azimuth and elevation is one coordinate system which can be used to describe the solar position relative to a human observer. It is a very natural coordinate system, because it puts us as the observer into the center. The azimuth is the direction in which we see the Sun, rising in the east ($\psi \approx 90°$), moving via south ($\psi = 180°$) and setting in the west ($\psi \approx 270°$). Please notice that observers in the southern hemisphere have a different view.

Equatorial system

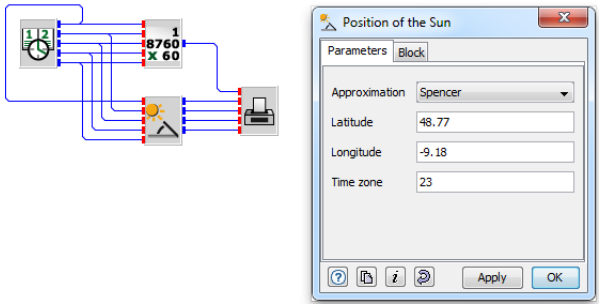
The SUNAE block also outputs the solar position in a second coordinate system, which uses declination δ and hour angle ω as coordinates.

To understand this coordinate system is a bit less intuitive. But imagine to be located at the center of the Earth, with the Earth as a globe made of glass with a line grid for the latitudes and longitudes on its surface. Every day the Sun

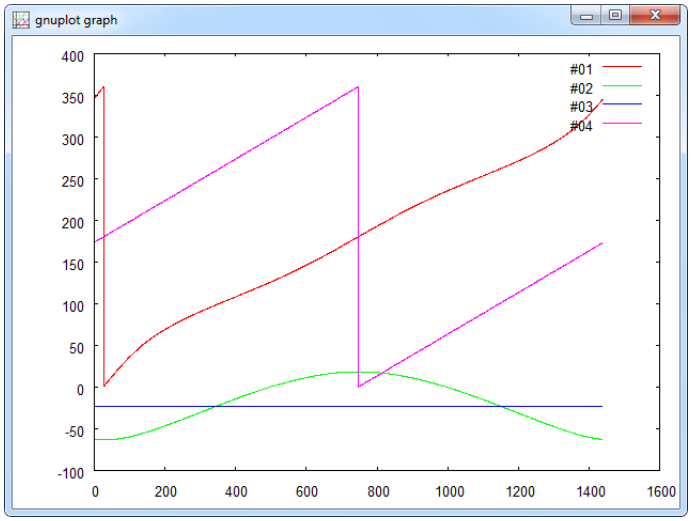
revolves once around this glass globe, following (almost) exactly a constant latitude. The angle between the equator and this latitude is called declination δ and is independent of any observer on the Earth's surface. We know that it varies between $+23.45°$ (our northern hemisphere summer) and $-23.45°$ (our winter).

The other angle, which describes the movement of the Sun around the Earth during a single day is the so-called hour angle ω. When a second observer is placed on the glas globe, his position and the moment when the Sun crosses this observer's longitude defines the hour angle $\omega = 0°$. Starting from here, the hour angle is counted positive as it follows the Sun on its way around Earth. The hour angle $\omega = 0°$ defines the true solar noon of the observer on the surface.

The following example shows the four coordinates for one day, 1 January 2012 at the location of Stuttgart, Germany.



This is the result.



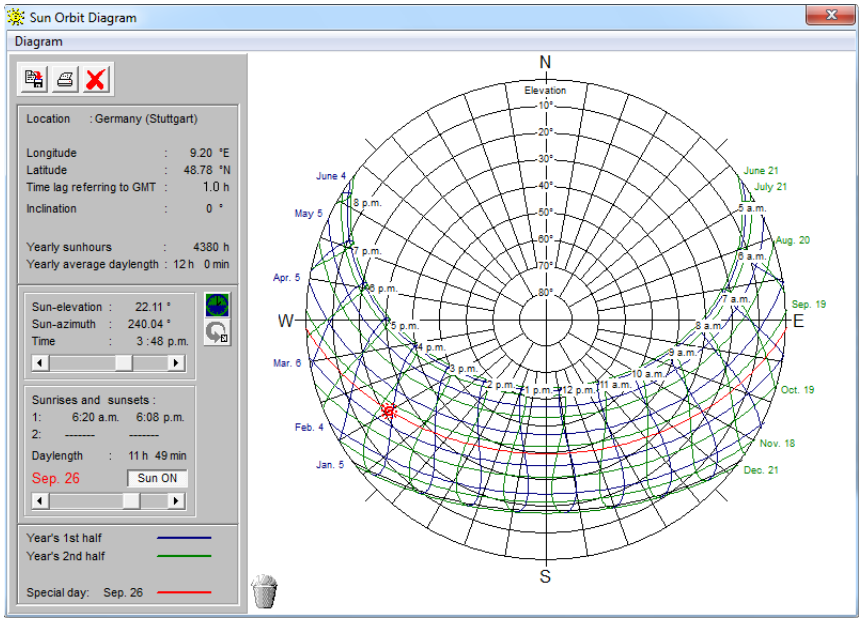
You find the SUNAE block in the Meteorology category under Geometry – Position of the Sun.

Three approximations for the calculation are currently implemented: The model of Spencer is the fastest in calculation time but the least accurate, the model of Holland and Mayer is a good compromise between calculation time and accuracy. The model of Michalsky is rather high in accuracy, it is the algorithm which is used in the astronomical almanacs.

With INSEL we could use the SUNAE block for the operation of a computer-driven pyrheliometer (a device to measure the direct solar radiation, which requires accurate two-axis tracking of the solar position), but this is beyond the scope of this Module.

SunOrb

If you are further interested in an understanding of the movement of the Sun, there is a nice tool called SunOrb that has been programmed at the University of Bochum, Germany in the group of Prof. Dr.-Ing. H. Unger. The program can be used to calculate and draw solar diagrams like the following one for Stuttgart.



You find SunOrb under the Tools menu or you can start it directly from the tool bar with a click on its icon .

Exercise

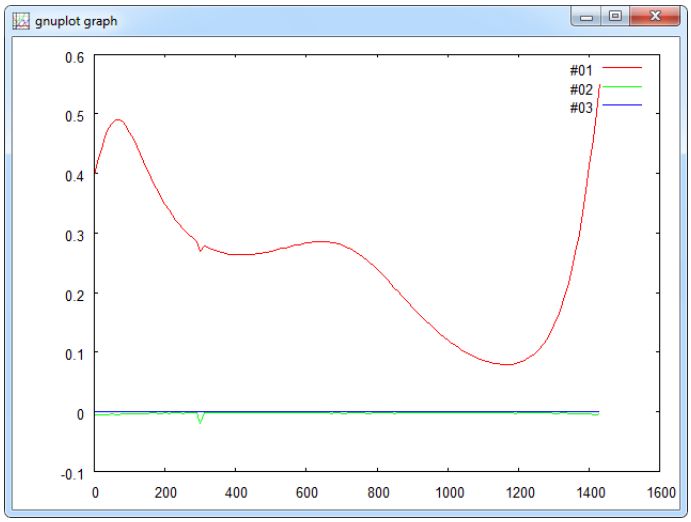
As an exercise with the SUNAE block you can compare the accuracy of the three models. (Hint: Use three copies of the SUNAE block, each with a different model. Take the Michalsky model as reference and calculate differences to the coordinates

of this model. If you follow this, you will need a Summation block SUM and a Change sign block CHS – you find both of them under the Math menu.)

Solutions

We provide four solutions in the `examples\tutorial\module2` directory in the files `sunae3a.vseit` (azimuth), `sunae3e.vseit` (elevation), `sunae3d.vseit` (declination), `sunae3o.seit` (hour angle).

As a result of the comparison for the azimuth angle, for example, we get this graph:



It shows the deviation of the azimuth angle calculation against the Michalsky model (red), the Spencer model (blue) overestimates the azimuth angle by up to 0.5 degrees on our reference day, 1 January 2002 in Stuttgart, whilst the deviation of the Holland/Mayer model (green) is almost not visible.

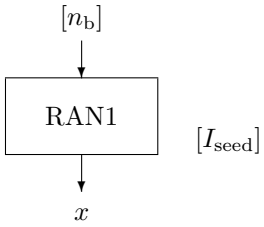
There are many more applications of the CLOCK block. We come back to some others in Part II.

Random numbers

Let us now turn our attention to the FDIST block, a Timer block which can be used to calculate the frequency distribution of any time series.

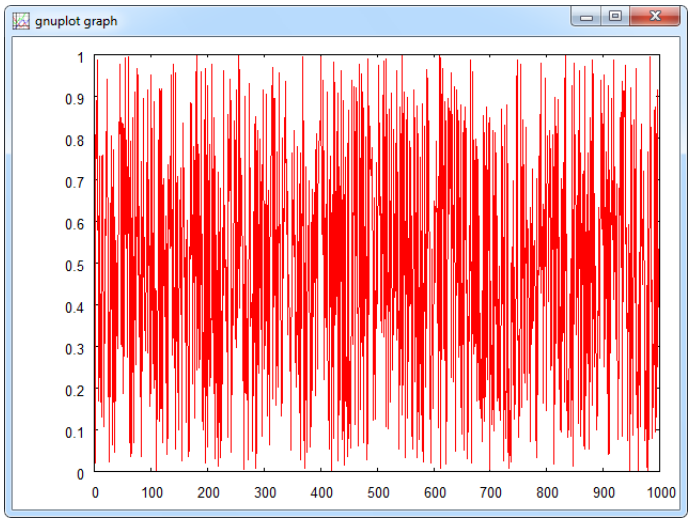
For the time series generation we use two random number generators, one that gives uniformly distributed, and one that gives normal or Gauss distributed random numbers. The first block is called RAN1, the second is called GASDEV, both are Standard blocks. You find these two blocks and the T-block FDIST in the Statistics category under Random numbers and Distributions, respectively.

We start with the RAN1 block for the generation of uniformly distributed numbers.



The block has an optional input, and an optional parameter I_{seed}, which can be used to initialise the block – different instances of the block can be used with different I_{seed} values and generate different time series, but all will have a uniform distribution. We start with 1000 numbers.

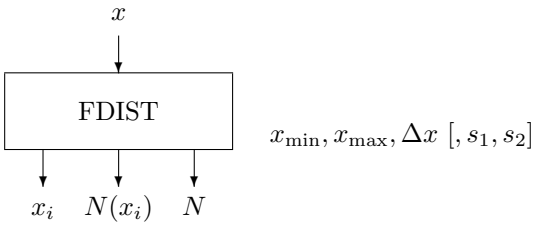
We have chosen a value of 1536 for I_{seed}. Please notice that the generated time series depends only on the I_{seed} value, so that the "random" numbers can always be uniquely reconstructed. The time series plot looks really random:



Difference between C-blocks and S-blocks

Maybe you have been wondering about the fact that we used the RAN1 block without an actual input. Since RAN1 is an S-block, it is automatically called by the inselEngine in every time step (of the main timer, which is the DO block in this case) of the simulation run. This is the main difference between C-blocks and S-blocks.

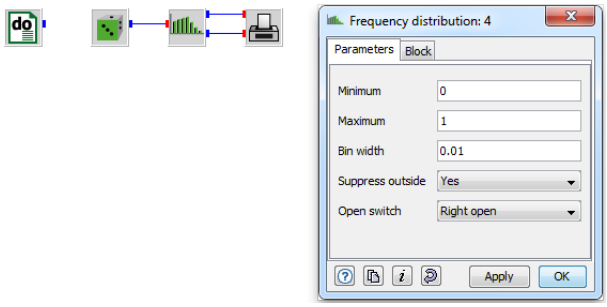
Now let's have a look at the FDIST block.



Frequency distributions

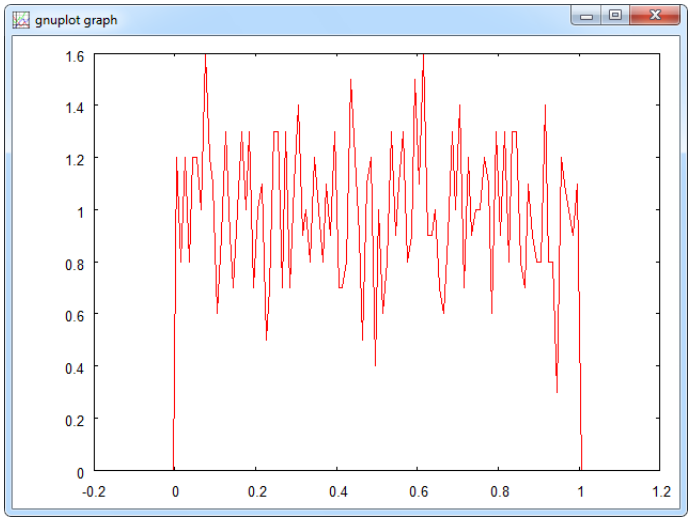
As expected, it has one input x for one value of the time series per call. Through the parameters we can fix the interval $[x_{\min}, x_{\max}]$ for the bins width Δx. Let us skip the optional parameters for the moment. Well, how do we expect the block to operate? We will deliver a time series (our RAN1 numbers) to the block's input. So far, so good. But when and how shall the block bring the results to the outputs?

Obviously, the block has to "wait" until the end of the time series to be then "informed" by the inselEngine that it is time to start the action, and that means: Play the role of a timer and deliver – one after the other – the x-coordinate x_i, starting with $i = 1$, the normalised number of data $N(x_i)$ in bin i. In addition, the total number of data N is an output which can be used for a the calculation of the absolute number data in a bin, for instance.

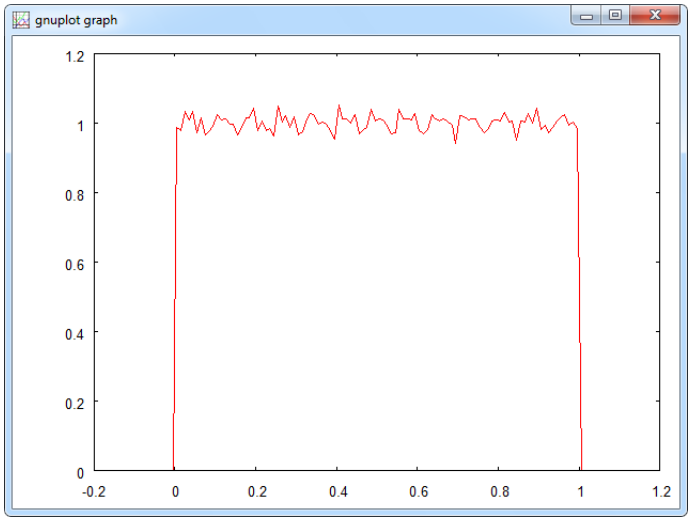


Due to the behavior of FDIST we can directly connect a PLOT block to the outputs of FDIST. Please notice that it is not necessary to connect the DO block with any other object. Due to its presence it generates a number of steps according to its actual parameter settings.

The result for 1000 numbers looks as follows.



For 100 000 numbers the distribution is already much smoother.



Exercise Plot the Gauss distribution on the basis of one thousand, ten thousand, one hundred thousand and one million normal distributed random numbers.

Solution See file `fdist.vseit` in the `examples` directory.

We could continue this Module with an extensive collection of examples for different blocks, but C-blocks are rather boring (and there aren't too many in INSEL), concerning T-blocks we have already looked at some important ones, and

S-blocks? Well, there are some hundred available in the different toolboxes. Perhaps, at this stage you should take your time and go through the block reference of INSEL. This will give you an overview on some of the basic blocks. When you do so, check the block group first and skip all non C-, T-, and S-blocks in the first run.

Questions

- There are seven different block groups in INSEL. Three of them have been discussed in detail: C-blocks, S-blocks and T-blocks. Can you explain in your words what the difference is?
- The PVI block has been used to calculate the characteristics of photovoltaic modules. Could you generate a graph which shows the I-V characteristic of a PV module for different module temperatures in one graph?
- When you create an INSEL block diagram INSEL sorts your model and creates a calculation list. Can you set up the calculation list for the example on page 36 that we used to show the four coordinates of the SUNAE block for one day, 1 January 2002 at the location of Stuttgart, Germany?