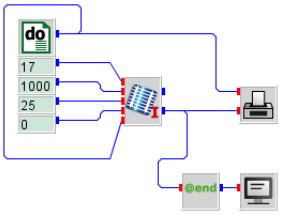
Module 4 :: If blocks

A programming language is not a programming language if it does not provide at least one statement which enables the use of an if-then-else structure. In INSEL this structure element is represented by the concept of If blocks, or I-blocks, in short.

In Module 2 we have already used an ATEND block as a first example for an I-block. Let us briefly recall its use. The block diagram which used the ATEND block was the following:



atendExample.vseit verschoenern

A DO block is used as timer which runs through one hour in steps of one second. For constant meteorological and operational conditions a PVI block calculates the warming up of a PV module from an initial temperature of 25 °C. The resulting temperature plot has been shown on page 26.

We saw from the graph that the module heats up to about 53 °C. What if we are only interested in the equilibrium temperature rather than the complete temperature profile? In this case we would like to let the module warm up, but display only the last, i. e. the equilibrium temperature value. This is a typical task for the ATEND block.

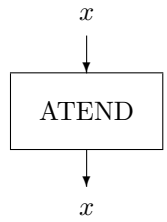
In the example, it uses the module temperature as input, whilst the output of the ATEND block – which is identical in value to its input – is connected to a SCREEN block. But the ATEND block ignores all input values until the simulation run is completed. Only then, the ATEND block lets the input signal pass.

Hence, from the point of view of the SCREEN block the SCREEN block is supplied by a value from the ATEND block only at the end of the simulation run, and thus displays only one value: the value at the end of the simulation run, which is the equilibrium temperature of the PV module in this case.

4.1 At end If blocks

Let us analyse the ATEND block in more detail.

@end



The block – like any other INSEL block – receives data depending on its input connection. But the ATEND block ignores all inputs until the end-of-run, i. e. until the condition whether the end of run is reached becomes true. Only in this case, the ATEND block lets the input signal pass through to its output and the inselEngine calls the blocks which are connected directly or indirectly to the ATEND block's output.

This is the typical behavior of an If block – it checks a specific condition. When the condition is true the blocks which make direct or indirect use of the I-block's output are executed, when the condition is not true the blocks which make direct or indirect use of the I-block's output are not executed.

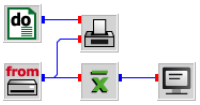
End of run

In case of the ATEND block the condition is the end of a simulation run. Other examples for blocks which use the end of simulation run condition are the blocks which calculate the average of an input signal over a complete run (block AVE), or cumulate an input signal over a complete run (block CUM), or find the absolute maximum (block MAXX) or minimum (block MINN) of a time series.

AVE block

We start with the average block AVE. In Module 3 we had used the file `meteo82.dat` which contains hourly records of meteorological parameters for the location of Oldenburg in Germany for one year. One variable of the time series saved in this file is the global irradiance on a horizontal surface in W/m^2. How can we calculate its annual mean value?

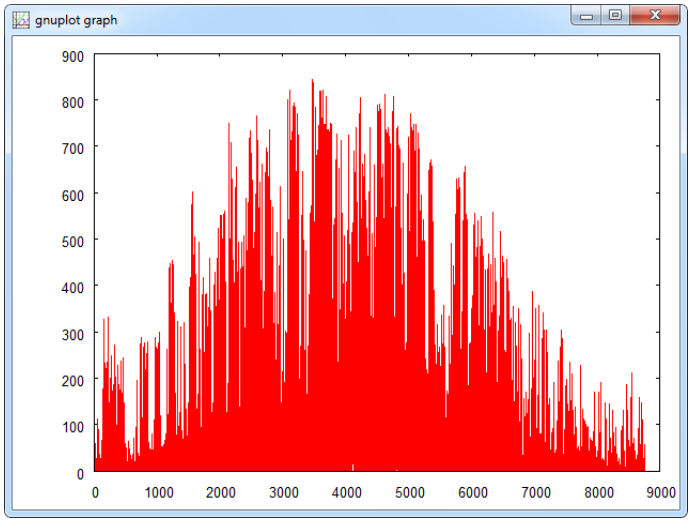
The answer is straight forward: Use an average block, to be found under the *Mathematics* – *Logics* category, connect it to the radiation output of a READ block which reads `meteo82.dat` and display the output of the AVE block with a SCREEN block. The block diagram is simple.



And the result is: The global irradiance on a horizontal surface in Oldenburg in the year 1982 has been 108.98 W/m^2 – did you find the same figure?

Please observe three details from our block diagram.

- We have added a PLOT block which displays a graph of the complete time series. The resulting plot is useful for a plausibility check that we have really configured our READ block with the correct global irradiance data.



- Since we are interested in the global irradiance data on a horizontal surface only, we skip all other data of the input file by using the format `(8X,F5.0,51X)`. Hence, our READ block has only one output.
- Again, the READ block is not connected to the T-block DO, but executed in each of the 8760 time steps (since READ is a Standard block).

The unit of the hourly irradiance data – and hence of the annual average as well – is given in $\mathrm{W/m^2}$. Physically spoken this is a power density, i. e. power in watt per area in square meter. There are some people in this world who seem to have slight problems with this kind of average calculation for solar irradiance, with somehow vague arguments like "But at night the Sun does not shine, so why shall I consider these hours in the calculation at all?" The answer is: The AVE block just calculates the global average $\bar{G}$ of the radiation time series $G(h), h = 1, \ldots 8760$ according to the standard definition of the average

$$\bar{G} = \frac{1}{8760} \sum_{h=1}^{8760} G(h)$$

Conversion of units

For those who prefer to think of global radiation as energy per square meter and time interval, it is easy to convert the annual mean value from $\mathrm{W/m^2}$ to $\mathrm{kWh\,m^{-2}\,a^{-1}}$. All we have to do is multiply $\bar{G}$ by the number of hours per year (which is 8760), and divide by one thousand for the conversion from Wh to kWh,

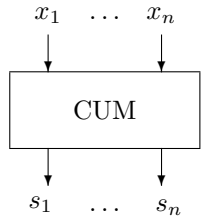
i. e. multiply $\bar{G}$, given in $\mathrm{W/m^2}$ by 8.76 to get $\bar{G}$ in $\mathrm{kWh\,m^{-2}\,a^{-1}}$. The result is easily calculated: $108.98 \times 8.76 = 954.66$ kWh per square meter and year.

If you like to use INSEL for the calculation, connect a GAIN block with parameter 8.76 to the AVE block's output and display the output of the GAIN block rather than the output of the AVE block directly. This is a simple example for the conversion of units with INSEL. Please notice that instead of using a GAIN block, it would have also been possible to use a CONST block with parameter 8.76 and a MUL block which multiplies the AVE block's output with the CONST block's output. The result is exactly the same but using the GAIN block saves one INSEL block in the block diagram and is therefore preferred.

There are three more If blocks available which are very similar to the function of the AVE block:

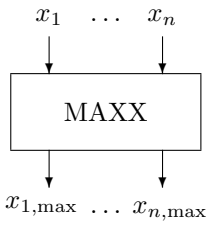
CUM block

The CUM block calculates the cumulated sum of its input over a complete simulation run.



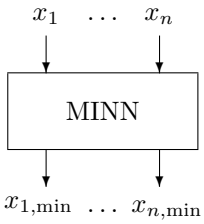
MAXX block

The MAXX block calculates the overall maximum of its input over a complete simulation run.



MINN block

The MINN block calculates the overall minimum of its input over a complete simulation run.

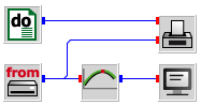


Exercise 4.1 Use the three blocks and apply them in order to calculate

- The cumulated value for the global irradiance on a horizontal surface in kWh/m^2
- The overall maximum value for the hourly global irradiance on a horizontal surface in W/m^2
- The overall maximum value for the ambient temperature in °C
- The overall minimum value for the ambient temperature in °C

as saved in file `meteo82.dat`.

Solution 2



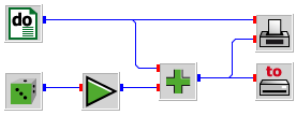
The maximum value of the hourly global irradiance on a horizontal surface is 843 W/m^2. ■

Fit blocks

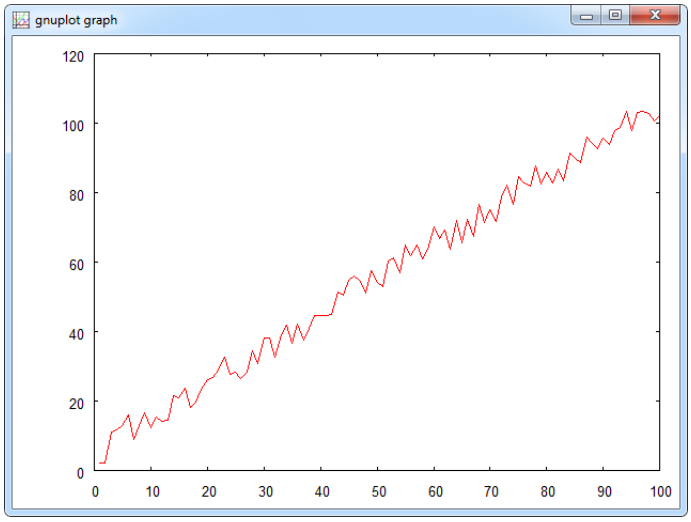
We turn our attention now to another set of If blocks which also use the condition end of simulation run and this is the set of fit blocks. What is a fit? A fit is a statistical method to approximate a set of data by an analytical equation of a given form. A very wide-spread and well-known fit uses the method called *linear regression*. In this case, the given (statistical) data set is approximated by a linear function. Before we discuss the FITLIN block let us create a data base for the function to be fitted.

`fitlin0.dat`

Let a DO block deliver 100 steps and a RAN1 block generate one uniformly distributed random number for each step. When we multiply the random numbers by a factor ten with a GAIN block, for instance, and add the DO block's output and the output of the GAIN block, we defined a scattered variable which can serve as data base for the FITLIN block. We have saved the data in a file named `fitlin0.dat` in the `examples\tutorial` directory. It has been calculated with this block diagram, saved as `fitlin0.vseit` in the `examples\tutorial` directory as well:



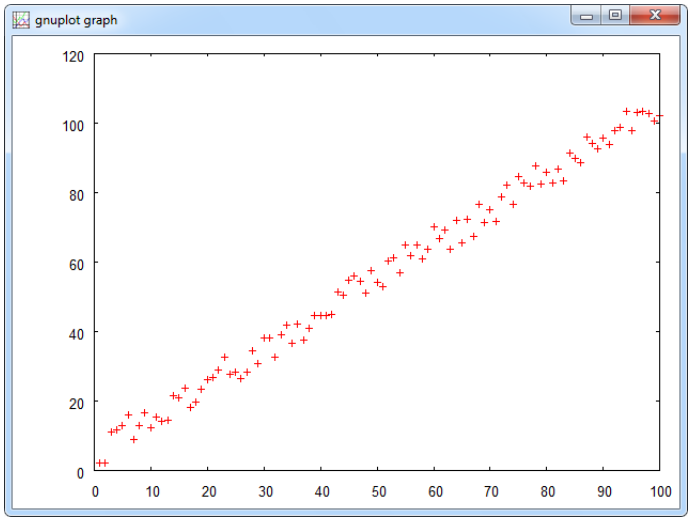
The resulting data look at least a bit scattered and show an obvious trend.



If you like, you can plot the data without the disturbing interpolation lines with Gnuplot.

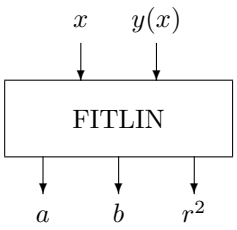
Hint Use Gnuplot in interactive mode, as briefly described in Module 3, page 67. Choose *Data Style – Points* from Gnuplot's *Styles* menu, and click the *Replot* button.

The result is this:



FITLIN block

Now we are ready to read the "statistical" data and perform a linear regression. The FITLIN block itself has the following layout:

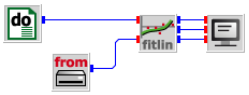


Two inputs must be connected and fed with data: An independent variable x, and an x-dependent variable $y(x)$. The block requires no parameters. Outputs are the variables a, b, and r^2, where a and b are the best approximations to the equation

$$y(x) = a + bx$$

and r^2 is the regression parameter which describes the accuracy by which the equation $y = a + bx$ approximates the data. $r^2 = 0$ is the case of absolutely no correlation, $r^2 = 1$ stands for the case where all data points are either absolutely correlated or absolutely anti-correlated.

The block diagram is simple again.

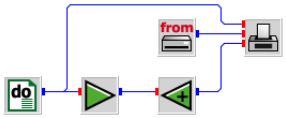


A READ block reads the scattered data from our file `fitlin0.dat`, just using the star format, for example. The FITLIN block finds the parameters a, b, and r^2. The SCREEN block displays the output with format (3F8.4). We get this result:

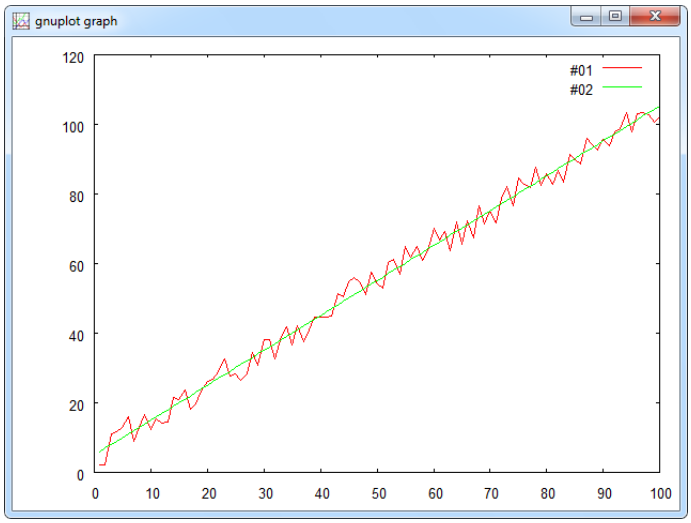
```
4.8319   1.0039   0.9903
```

Once the result is known ($a = 4.8319$ and $b = 1.0039$) we can plot the linear equation $y = a + bx$ and the scattered data in one diagram to see how good the fit is.

The block diagram which reads the scattered data requires only minor changes. A GAIN block is used to multiply the output of the DO block (the variable x) by the factor $b = 1.0039$, an OFFSET block with parameter $a = 4.8319$ and a SUM block builds the sum $y = a + bx$, which is displayed by the PLOT block.



This is the plot:



Standard fit routines

Some standard fit routines are available as blocks like block FITEXP, which fits data to the exponential function $y = a\exp(bx)$, block FITLN, which fits the logarithmic function $y = a + b\ln(x)$, and block FITPOW, which fits the power function $y = ax^b$.

There are some much more sophisticated fit blocks available in INSEL. For example, the PVFIT1 and PVFIT2 blocks are used to fit data which describe the

performance of photovoltaic modules to equations known as the one-diode model and the two-diode model. For further details on these blocks please refer to the respective reference manuals.

An example for the PVFIT2 block will be presented in Module 7, page **??**.

4.2 If blocks with a parameter

AVEP block

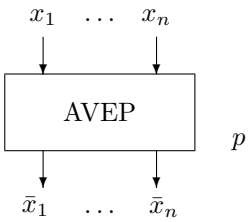
During the discussion of the average block AVE we have seen a plot of the hourly time series of global radiation for Oldenburg, Germany. There was hardly something to distinguish, since the plot was basically a lot of red ink.

Annual radiation time series are much better visualized as series of daily data rather than hour by hour. An alternative would be a carpet plot – see block PLOTPMC for further details.

In order to calculate the daily means an average block would be useful which cumulates the radiation data over one day, i. e. 24 hours, divides the cumulated sum by 24, and outputs the result after every 24 hours. This is exactly what the AVEP block does – it calculates an average over a number of steps as specified by a parameter p.

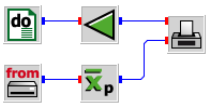
$$\bar{x} = \frac{1}{p}\sum_{i=1}^{p} x_i$$

which is exactly the same definition as the formula used by the AVE block, the only difference being that p is a free block parameter.

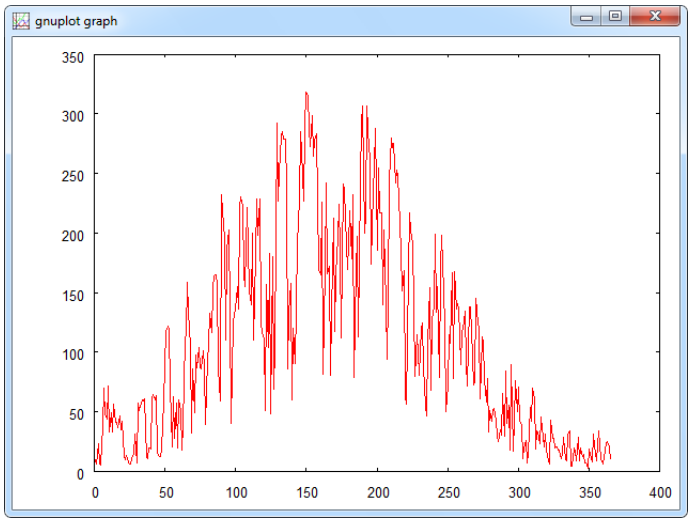


There are some very similar INSEL blocks named CUMP, MINNP, and MAXXP. You can probably guess what their functions are – check the Block Reference manual for details.

Let us construct a simple application for the AVEP block, and plot the time series of daily global irradiance data on a horizontal calculated from file `meteo82.dat`.



From a previous example we have simply replaced the AVE block with an AVEP block, used an ATT block for the division of the hours by 24 and plot the time series of daily data.



Please notice again, that the daily averages value are given in W/m^2. Since the time step of the data is one hour we can interpret the radiation data as Wh/m^2 as well. If you prefer to display the radiation data in $kWh\,m^{-2}\,d^{-1}$ you should multiply the values by 0.024. Since the maximum daily value is about 333 W/m^2 this corresponds nearly 8 $kWh\,m^{-2}\,d^{-1}$ in summer.

Hint

Do not connect the outputs of the four different If blocks to one SCREEN block – INSEL will not accept this (try it) and display an error message that the SCREEN block depends on not enclosed If/Timer-blocks.

Bug or feature?

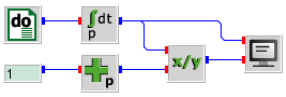
This behavior has been newly introduced since version 6.0. Whether it is a feature or a bug is still not clear – most probably it must be considered as a bug.

The background is that different I-blocks can have different conditions. Hence, depending on the conditions some unwanted effects might occur if outputs of I-blocks with different conditions are brought together. But in cases like the one we are discussing, when the conditions are all the same – end of the simulation run – it should work. But it doesn't. The way out is to use four SCREEN blocks for the four outputs.

Workaround

In case you wish to use an averaging block and a cumulation block and write the results to a data file, the above mentioned behavior does not allow this. However, a workaround to use the cumulation block and "simulate" the average block by SUMP block which cumulates constant 1 values with the appropriate parameter p

and divide the cumulated signal by the output of the SUMP block:



Letting the DO block count from 1 to 10 and setting the parameters of CUMP and SUMP to 10 leads to the expected result:

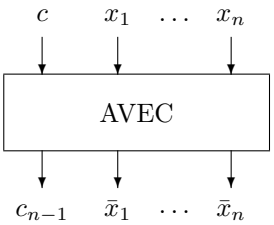
```
55.000000       5.5000000
```

4.3 Conditional If blocks

What, if we want to calculate monthly means rather than daily? The complication is, that days always have 24 hours, but the number of days in a month is not constant. For example, January has 31 days, February 28, or – if it is a leap-year – in February the number of days is 29, March has 31 days and so forth.

AVEC block

A block which solves this problem conveniently is the AVEC block (average with condition). The layout of the block as follows.

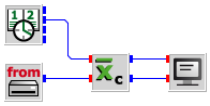


The block has two inputs: a condition input c and a signal input x_i. The idea of the block is to collect input data x_i as long as the condition input c remains constant. When the value of c changes, the block calculates the average over all x_i where c has been constant and outputs the average value. Let us look at an example first and then understand some more details about the block.

Monthly means

Assume, that we want to calculate the monthly mean ambient temperature values from the hourly data as stored in file `meteo82.dat`. Since the calculation of the average depends on the Gregorian calendar it is obvious that we use a CLOCK block as timer which runs through the hours of the year 1982.

For every time step of the CLOCK block INSEL reads one record from the file. The inputs to the AVEC block then are (i) the condition input month M as given by the CLOCK block, and (ii) the data input T_a from the READ block. The following block diagram does the job.



This is the result:

```
1.0000000     -0.59731185
2.0000000       1.3403274
3.0000000       4.9831991
4.0000000       7.1958332
5.0000000       11.912768
6.0000000       16.160418
7.0000000       18.441263
8.0000000       17.304436
9.0000000       15.336250
10.000000       10.365457
11.000000       6.6379166
12.000000       2.4293010
```

When you remember the file format of `meteo82.dat` as discussed in detail in Module 4, page 53 the ambient temperature is the tenth parameter of the file, so that we used the format `(33X,F5.1,26X)` and a READ block with one output to read the temperature time series.

A detail

Please observe that the SCREEN block uses the condition output of the AVEC block rather than the month output of the CLOCK block to display the monthly mean temperatures.

Why? Try, and figure out the reason by yourself for a moment.

Well, what happens? The CLOCK block starts with the first of January, zero hours, the READ block reads the corresponding data record, the AVEC block receives the data, and this sequence continues, continues ... All the time the condition input of the AVEC block is equal to one, i. e. the AVEC block remains in data collection mode.

Then comes the last hour of January. The output of the CLOCK block is year 1982, month 1, day 31, hour 23 (not 24!). From the point of view of the AVEC block nothing special happens – the condition input is still equal to one, i. e. the block remains in data collection mode.

But then: in the next time step the CLOCK block changes its outputs to year 1982, month 2, day 1, hour 0 (not 1!). Now, from the point of view of the AVEC block, the condition input (month) has changed, i. e. the block has to perform some action.

The AVEC block calculates the average value, prepares the calculation of the average for the next condition (which is February, logically), outputs the monthly

mean value for condition $c = 1$ (i. e. January) and request from the inselEngine to execute the successors – which is the SCREEN block only, in this case.

How shall the SCREEN block know that the value it gets is the January value? The output of the CLOCK in the actual time step says 2, i. e. February already. This is the reason why the AVEC block outputs the average value and the corresponding condition coordinate.

A second thought

Did you recognize that it is in fact a problem to display the last mean value?

The last time when the AVEC block is year 1982, month 12, day 31, hour 23. In this step, no change in the condition happens, and hence the AVEC block cannot know that the simulation run is finished.

Destructor call

For such cases INSEL has a mechanism that all blocks receive at least one additional so-called destructor call. From the AVEC block's point of view this implies a definitive condition change. This is the last chance for the AVEC block to calculate the last average value and put it on its output.

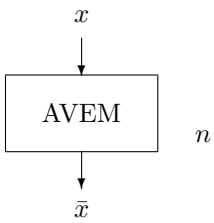
The same mechanism applies to the PLOT block. Maybe now you can have a better understanding of the details about the PLOT block discussed in Module 3, page 65.

More conditional If blocks

There are some more I-blocks which use a condition input, like CUMC, MINNC, MAXXC. Please check the Block Reference manual for further details on these blocks.

AVEM block

Another block which is closely related to the mentioned ones is a block named AVEM which calculates a moving average of a given time series. The name might indicate that the AVEM block is another example for an If block, but actually the moving-average block is a Standard block.



The AVEM block calculates its output from a connected time series by the formula

$$\bar{x}_j = \frac{1}{\min\{n, j\}} \sum_{i=\max\{1, j-n+1\}}^{j} x_i$$

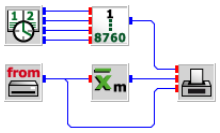
which means that for any time step the AVEM block provides an average value over the previous time steps as defined by the block's parameter – let us neglect

the initialization problem for the time being. This means, that the AVEM block outputs a value for each time step. But this is the behavior of a Standard block which always outputs a value, whenever it is called.

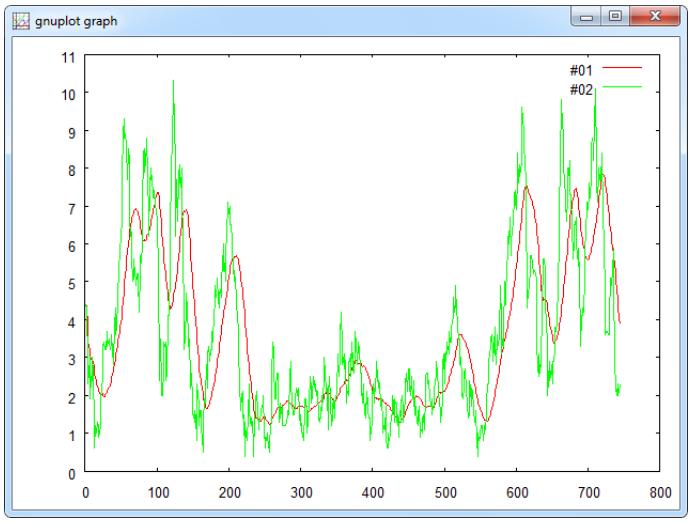
What makes the difference to If blocks is, that If blocks provide output values only under certain conditions and request from the inselEngine to execute the successors only now and then, depending on their condition.

As an example for the AVEM block we calculate the moving average of the wind speed data for January as saved in file `meteo82.dat`. As interval for the calculation of the moving average we use 24 hours. Remember that the wind speed is the last value in the records with format `F5.1`.

This is the block diagram



and this is the resulting plot:



Please observe that the AVEM block smoothens the high fluctuations in the hourly wind speed data – as expected.

Side remark about wind directions

Allow us a last short remark on the AVEM block. Maybe you have the idea to look at the moving average of the wind direction time series. There is a little problem in doing so. Did you notice, that when you average wind directions from

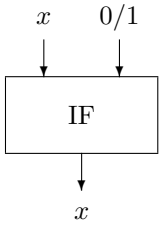
North that it may happen that the average of something like North-North-East and North-North-West must be calculated?

North-North-East direction corresponds to around 350 degrees and North-North-West to around 10 degrees. The average is 180 degrees, hence South direction, which is obvious nonsense. But we do not further look at this aspect here – one reason being that the AVEM block is not even an If block but a Standard block.

4.4 General if conditions

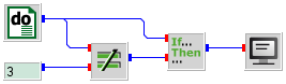
"The" If block

We turn our attention now to the more general cases of if conditions. The most natural candidate for a block of the group of If blocks is a block which gave the group of I-blocks its name: The block named IF.



This block has two inputs, x and a logical input which can be either zero (false) or one (true), and one output – the signal that is connected to the x input. The IF block lets the input signal pass through, if the second input – the condition input – is true, otherwise it doesn't. "Otherwise it doesn't" means, the output is not available in the current step, and hence, the successors of the block, i. e. all blocks which make direct or indirect use of the IF block's output get no signal and are therefore not executed.

The best way to illustrate this behavior is a simple example. Let us construct a filter which lets all numbers pass except the number three.



The DO block counts from one to five, i. e. its parameters are set to 1 for the initial value, 5 for the final value, and 1 for the increment. The CONST block uses a value 3 as parameter. The block with the symbol ≠ is the NE block (not equal) and checks whether its two inputs are not equal (true) or equal (false). The NE block is a Standard block and can be found in the *Mathematics – Logics* category.

Both, the output of the DO block and the output of the NE condition block are connected to the IF block. Finally, the IF block lets all values pass through,

except the value 3. So, from the point of view of the SCREEN block, which is connected to the IF block, the SCREEN block is served with data except when the output value of the DO block is equal to 3.

Test it! What do you expect to see as output? The values 1, 2, 4, and 5. Test it, please.

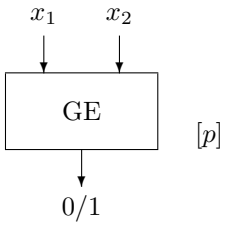
Logical conditions Like in any ordinary programming language the problem to set up an if structure is to formulate a condition which evaluates either to true (1) or false (0), and execute the if branch when the condition is true or to perform no operation if the condition is false.

INSEL provides blocks for the formulation of all standard logical conditions. These standard conditions are

- Equal (block EQ)
- Not equal (block NE)
- Greater than (block GT)
- Greater or equal (block GE)
- Less than (block LT)
- Less or equal (block LE)
- And (block AND)
- Inclusive or (block OR)
- Exclusive or (block XOR)

All of them are Standard blocks and with these blocks a lot of logical conditions can be constructed. The functions of the different blocks should be self explaining.

GE block But let us look at the example of the GE block which checks for a greater-or-equal condition.



As expected the GE block has two inputs x_1 and x_2 and checks whether x_1 is greater of equal x_2. If yes, the block outputs a one, otherwise it outputs a zero. We have added an optional parameter p which weakens the hard equal condition.

Absolute equity? What is the reason? With INSEL we are doing numerics mainly on the basis of

Fortran REAL variables. These variables in the computer's memory have a rather limited accuracy of about seven to eight significant digits. Hence, comparing them to being absolutely equal might lead to unwanted results. Therefore, with the GE block the variables must not necessarily be absolutely equal but can differ by a tolerance p and are still considered equal by the GE block. When p is not specified, the GE block goes the hard way and compares for absolute equaltiy.

If some of the other condition blocks should be unclear, please refer to the Block Reference manual for the details.

4.4.1 Load profiles

In the next step let us practice to formulate if conditions for a realistic example. One of the most natural applications of the condition blocks like EQ, GE, GT, etc. is the formulation of load profiles in the widest sense.

Exercise 4.2 Let us assume we want to construct a condition for a public building – a library, for example – and we want to decide whether it is open (true) or not (false). First we have to define the hairy details.

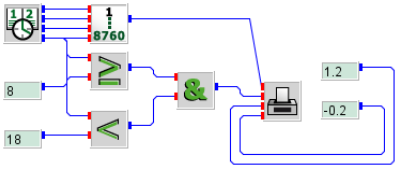
For reasons of simplicity, let us assume a not-too-complicated opening schedule. Let our library be open every day from 8 a.m. to 6 p.m. except the weekends, i. e. Saturday and Sunday, when our building is closed.

Try, and solve this problem as an exercise.

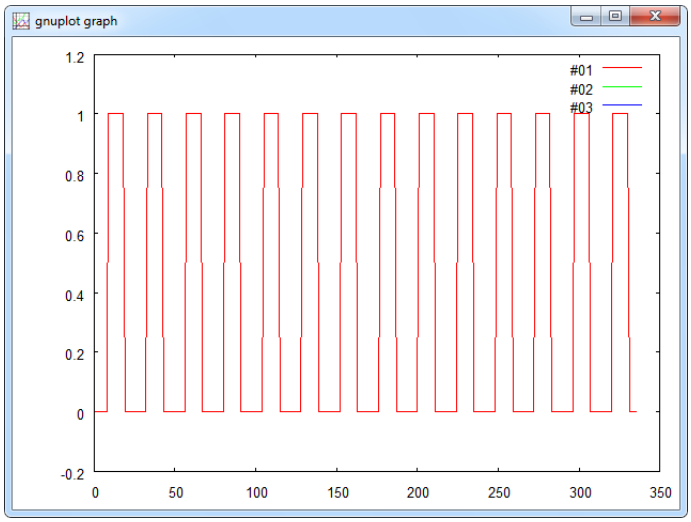
Solution Our solution process goes like this: At first, we ignore the complication of the weekend closure. Obviously, we will use a CLOCK block. The hour output of the CLOCK block will be used to decide whether it is already opening time or closing time. For sure, we need two constants for the opening time (8 o'clock) and the closing time (18 o'clock).

For the first step, we then need a GE block, a LT block, and an AND block to formulate our simplified condition. Please notice, that an LE block in combination with a constant 18 would keep the library open until 7 p.m. Do you copy?

Then we plot the opening condition in order to check whether our simplified solution works or not. If not, we go back and make changes until the simplified solution works. Our preliminary solution looks like this:

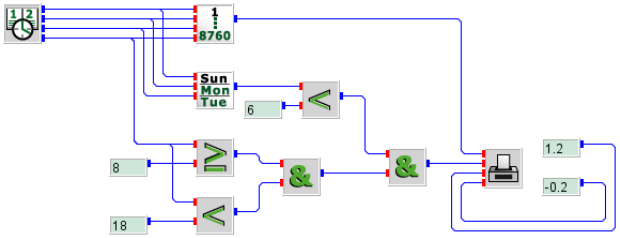


We have added two constants in order to make the plot a little nicer. The opening hours indicator for the first two weeks of January 2012 looks like this:

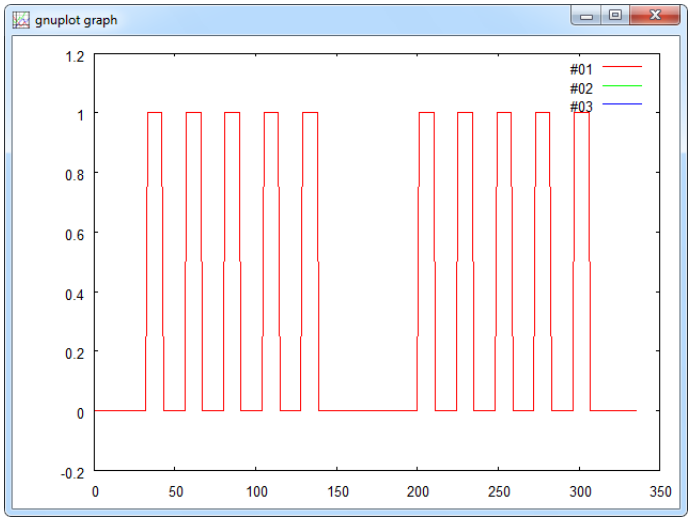


DOW block

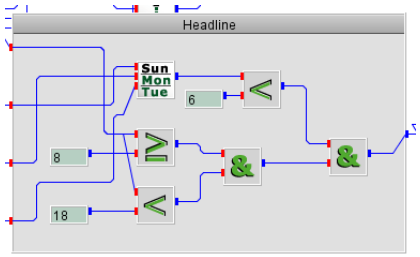
The last thing to complete our solution is to sort out the weekend case. In order to check for the day of the week we can use the DOW block which uses the a Gregorian date as input and returns a one for Monday, a two for Tuesday, and so forth. So, for our opening indicator we can check whether the DOW output is less than six – i. e. the library is open, or not. This makes a minor modification to our previous block diagram.



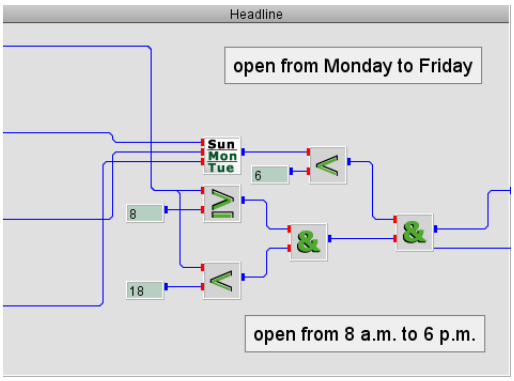
The opening scheme now looks as follows:



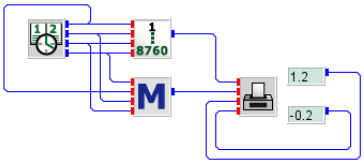
Macro This is the first place to use a macro an encapsulate the logic scheme. This makes the block diagram easier to read. Still a bit scrambled rooting.



In addition we can add some labels to our first macro which make it easier to understand what we did.



Finally, our block diagram reduces to



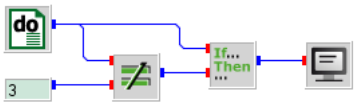
Exercise 4.3 As an exercise, add a modification to the opening schedule such that the library is closed during August an plot the annual opening scheme.

Blocks IFPOS and IFNEG

There are two If blocks named IFPOS and IFNEG which should perhaps be mentioned here, because they cover two rather common filters for (strictly) positive and (strictly) negative numbers. Their function and use are probably self explaining, if not, please check the Block Reference manual for further details.

4.5 Calculation list

Let us understand the concept of If blocks a little deeper by looking again at a previous example:



This model includes five INSEL blocks, namely the T-block DO, the C-block CONST, the S-blocks NE and SCREEN, and last but not least the I-block IF. Let us answer the question how exactly INSEL converts this block diagram into a calculation order.

As a general rule INSEL checks at first whether there are C-blocks included in the model. In this case INSEL finds exactly one C-block, namely the CONST block. INSEL "sorts" this block into the first place of the calculation list – we have already seen an example of a calculation list in Module 2, page 29.

Then INSEL looks for T-blocks, finds exactly one in the model, namely the DO block, and sorts the DO block into the second place of the calculation list. In the third step INSEL looks for S-blocks in the model. In this case there are two: the NE block and the SCREEN block.

Known inputs

As mentioned earlier, INSEL can execute blocks only, when their input signals are already "known", which means that they have an actual value. The "known" signals are all outputs of blocks in the calculation list so far, in our case this is the constant value of the CONST block and the output of the DO block. Hence, it is possible to sort the NE block into the third place of the calculation list. Please

observe, that there is no way to sort the SCREEN block into the calculation list so far, since its input signal is not yet known, because the IF block does not yet appear in the calculation list.

There are no more S-blocks to consider in this example, so INSEL checks for blocks of other groups and finds the IF block. Since both its inputs are known already INSEL sorts the IF block into the fourth place of the calculation list. Now all blocks which make use of the IF block's output are analysed – in this case the only left block is the SCREEN block, whose input is now known and can be sorted into the calculation list.

As a result, INSEL found the block order CONST, DO, NE, IF, and SCREEN.

It is now obvious that the CONST block is executed first. Due to the function of the block the constant parameter of the CONST block is connected with the blocks's output, that's all. The next block to call is the DO block, which connects its initial value with the block's output. Then the NE block compares its first and second output (not knowing where the values come from). If they are different, the NE blocks writes a 1 (logical true) to its output, otherwise a 0 (logical false).

Jump parameter

Next, the IF block is called. Two different things can happen: Either the second input is equal to 0, then the successors of the IF block are skipped, or the second input is equal to 1, then the successors of the IF block must be executed. During the sorting routine INSEL found that there is exactly one successor of the IF block, namely the SCREEN block. Usually INSEL jumps one step to the next block in the calculation list to find the next block to be executed, but after the IF block is executed INSEL needs to jump either one step to the SCREEN block, execute it, i. e. display the input on the monitor and then reach the end of the calculation list or jump two steps and skip the SCREEN block and reach the end of the calculation list.

The decision is made by the IF block, which is the only candidate who knows the meaning of its second input. The IF block informs INSEL what to do next, by setting the so-called Jump parameter either to 2 (skip the next block in the list) or 1 (execute the next block in the list)

When the end of the calculation list is reached, INSEL looks backward in the calculation list to find the next T-block and give control to it, which means that the DO block will increase its output by the increment defined as the block's third parameter and the next block is the DO blocks successor in the calculation list, i. e. the NE block. Please notice, that the CONST block will never be reached due to the calculation list rules.

The algorithm is executed until the DO block has "fired" all its values, then on the next call the DO block informs INSEL that nothing is left to do and INSEL ends the program.

We can summarize the discussion with a last look at the calculation list including the block names, block groups and Jump parameter values of each block:

```
Number    Block       Group    Jump
-----------------------------------
     4    CONST           C       1
     5    DO              T       1
     2    NE              S       1
     1    IF              I      -2
     3    SCREEN          S      -3
-----------------------------------
```

Forward jumps

Please observe, that rather than pointing to the end of the calculation list the Jump parameters point to the block which has to be executed next. So – although the IF block has a negative parameter in this example – I-blocks are characterised by the property that they allow forward jumps in the calculation list.

Nested If blocks

In the discusion of timer blocks we have seen that T-blocks can be nested. It is also possible to nest I-blocks, but it is time for a break and we postpone this topic for the time being.

Exercise 4.4 Calculate the annual mean ambient temperature as stored in file `meteo82.dat`.

Exercise 4.5 Plot the daily mean ambient temperature as stored in file `meteo82.dat`.

Summary

- You have learnt that If blocks – or I-blocks, in short – can be used to skip execution of blocks which are directly or indirectly connected to I-blocks.
- Some typical examples for blocks of the I-group are blocks which calculate averages or cumulative sums, for example.
- There is a set of blocks which perform numerical fits to statistical data like the linear regression block FITLIN, for example.
- A block named IF allows for the definition of practically arbitrary conditions.
- There are blocks like EQ, NE, etc. which can be used to construct general conditions from very simple to very complex.