EuroOffice Solver

EuroOffice Linear and Non-Linear Solver is an EuroOffice extension by Multiráció Ltd. (Budapest, Hungary).

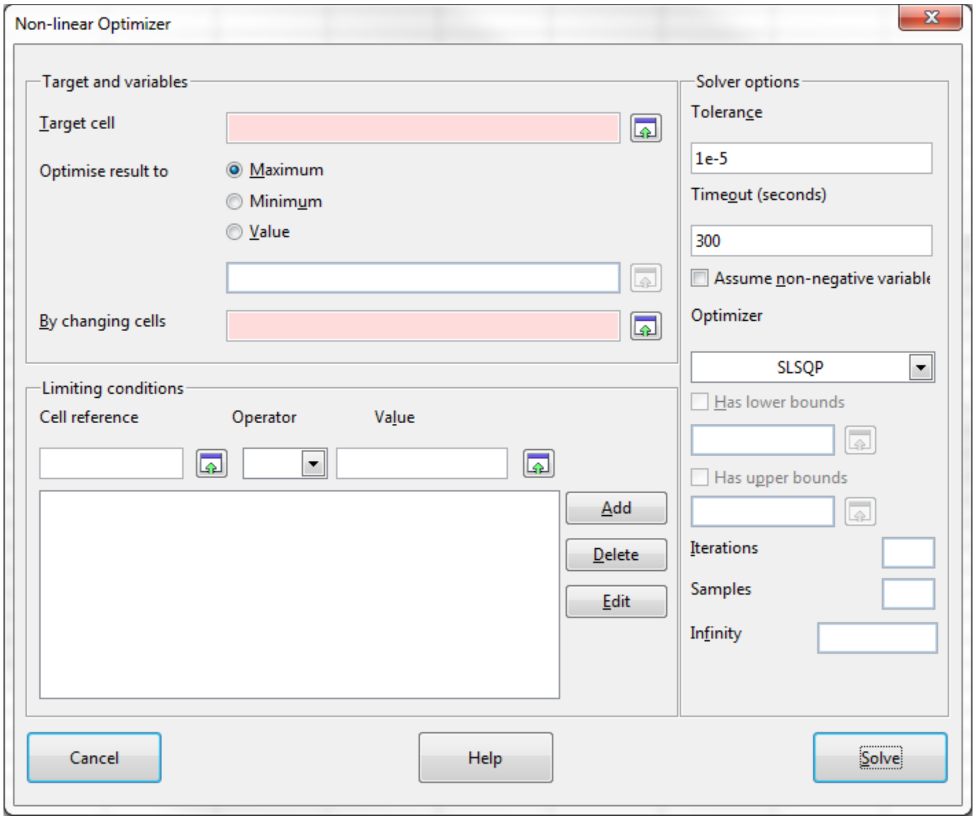
The built-in solver engine in OpenOffice is rather limited since it allows one to solve only linear problems. EuroOffice Solver is an improved solver engine which allows you to solve both linear and non-linear optimization problems either unconstrained or with linear or non-linear constraints.

Getting started

After installing the .oxt extension, the solver engine provided by the extension must be selected in the solver dialog (accessible via the Tools -> EuroOffice Solver menu) Click on "Options" and choose the solver engine "EuroOffice Solver". This dialog also allows one to change settings.

Setting up the model

In order to find those variable values of an equation which result in the optimal value in the target cell, that is the "objective", arrange your data on a spreadsheet and fill in the cell references in the dialogue box. After clicking on the Solver menu item, the Solver dialog appears.



Non-linear Optimizer

Optimise result to: ● Maximum ○ Minimum ○ Value

Tolerance: 1e-5

Timeout (seconds): 300

□ Assume non-negative variable

Optimizer: SLSQP

□ Has lower bounds

□ Has upper bounds

The controls in the dialog are as follows:

Target cell

This contains the value of the objective function which depends on the "variables" (see below). On the right hand side, you can use the shrinking cell reference boxes for easier data input.

Optimize result to

Here you can choose whether you want to minimize or maximize the target cell or you want to set it to a fixed value.

By changing cells

This contains the cells that should be changed, that is the "variables" whose values the value in the "Target cell" depends on. If there are more than one changing cell, this may be a cell range.

Limiting conditions

Here you can specify the constraints of the problem (if there are any). Both the left and right hand side of the constraints can be either a cell or a cell range - in the latter case the length of the ranges must be equal to each other. Constant values should be always written on the right hand side of the constraint. The Operator can be chosen to be the following: <=, =, >=, Integer, Binary. If the latter two are chosen, the Value field should be empty. The Integer conditions mean that the variables set here are integer numbers, while the Binary means that they are TRUE (1) or FALSE (0).

Tolerance

This is the tolerance condition for terminating the calculation, that is the calculation would stop if the target value would not change within the interval specified here. If this value is too small then you might not get the preferred results, so if the simulations gives odd results then you should run it again with a different value of this parameter.

Timeout in secs

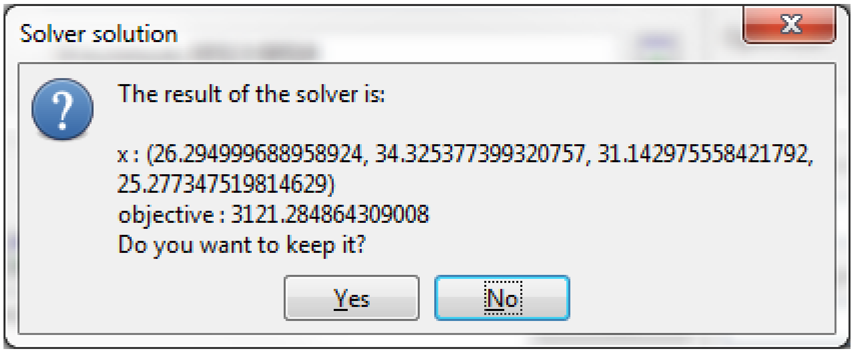
Some problems might take some time to calculate. If you want to wait only for a limited amount of time, you should set this parameter accordingly.

Assume Non-Negative Variables

If all the variables must be positive then we can specify this by checking the checkbox here. This way the constraints that the variables must be greater or equal to zero may be omitted from the set of constraints.

The calculation

If everything is set up correctly then the calculation can be started by pushing the **Solve** button. Depending on the model, after a few seconds the program will stop, and if a solution is found, a dialog will appear, which looks like this:



The result of the solver is:

x : (26.294999688958924, 34.325377399320757, 31.142975558421792, 25.277347519814629)
objective : 3121.284864309008
Do you want to keep it?

If you decide to keep the result then press the **Keep result** button, otherwise push the **Restore previous** button. In the latter case, the starting values of the optimization variables will be restored in the spreadsheet.

General tips

- If there are no constraints given then the solver engine will use the Nelder-Mead method, otherwise the SLSQP method is used.

- If you think the result is not adequate then please change the starting values or the error limit.

- If the model contains a lot of constraints, or there are a lot of changing cells, the program tends to be slow. Keep this in mind and set **Timelimit** accordingly.

- In case the **Integer values** condition is chosen, the program will calculate as if the problem was real valued and then it will use the usual rounding rules to obtain the final results. This may lead to plus or minus one differences in the obtained result as compared to the correct result.

- In some cases you may observe a warning message. This is an issue with the SLSQP algorithm and it can be caused by a lot of factors. Nevertheless you may chose to keep the results anyway or you can restore the original values and then you may try to recalculate the problem with a different error limit.