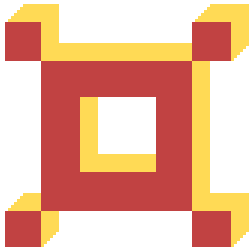
# User Manual
# xyzOPT V1.0



Dipl. Math. Annette Mura
info@xyzopt.com

March 7, 2023

# Contents

# 1 Enjoy your time with xyzOPT

xyzOPT is an application from the field of operations research.

xyzOPT makes it possible to model certain mathematical optimization problems with drag and drop. It is suitable for modelling many problems being mixed integer problems (MIPs).

xyzOPT itself can be used to model proplems not to calculate a solution. For this purpos mip-solvers are integrated - at this stage of development CBC by Coin-Or can be used.

xyzOPT is suitable for beginners and specialists in the field of optimization theory: beginners can model problems by drag and drop in case specialists constructed the required model building blocks before.

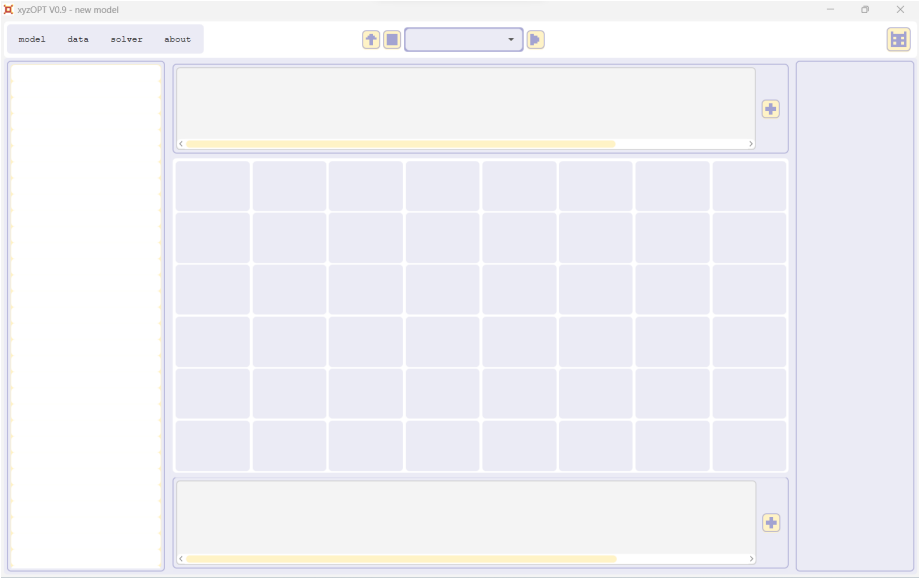
xyzOPT implements an interface to read and write data from spreadsheets.

xyzOPT models can be saved as xml.

xyzOPT has been developed in Java and was tested on Windows 11. Since it's a Java-application it might also run on other platforms. This wasn't tested yet.

**Have fun with xyzOPT!**

# 2 The user interface




xyzOPT distinguisches between model building blocks and model items: the model building blocks are templates for constructing model items.

The model building blocks are defined by mathematical formulas, which can be described within the model editor using the native algebraic modeling language. The model editor is opened by clicking a model building block.

Each model item is characterized by its model building block and input data can be entered.

## 2.1 The menu bar

The menu bar is positioned on the upper left.

### 2.1.1 Model

To create, to open or to save a model the item model is selected.

### 2.1.2 Data

Via the item data input data and results can be configured.

### 2.1.3 Solver

Via the item solver the mip-solver can be chosen.

#### 2.1.4 About

The item about contains information on the user rights and the user manual.

### 2.2 The navigation bar

The navigation bar is positioned in the upper middle.

#### 2.2.1 Up button

To change the grid content the up button can be used.

#### 2.2.2 Save button

To save the current model the save button can be used.

#### 2.2.3 Selection box

Via the selection box the problem to be optimized is chosen.

#### 2.2.4 Run button

Via the run button the optimization is started.

### 2.3 The waste bin

The waste bin is located in the upper right.

### 2.4 The model tree box

The model tree box is located on the left.

Each model is structured as a tree in graph-theoretical sense. The model items are the nodes of the tree, the root node is hidden.

### 2.5 The grid

The grid is located in the center.

The grid pictures an inner node of the model tree in detail.

### 2.6 The model box

The model box is located above the grid.

New model building blocks can be created and configured here.

## 2.7 The data box

The data box is located on the right.

It dispays the data of the model items.

## 2.8 The sheet box

The sheet box is located in the bottom.

Here it is possible to connect input data and results with spreadsheets.

# 3 For beginners: modeling with drag and drop

This chapter focuses on using already defined model building blocks. From each model building block model items can be derived with drag and drop and its data can be specified. Is the modeling of a problem completed, the problem can be optimized.

## 3.1 Derive a model item with drag and drop

To derive a model item select a model building block (drag) and drop it within the grid.

## 3.2 Specify the data of a model item

First the model item is to be named.

### 3.2.1 Explicit data definition

Indexsets can be defined explictly using the form $\{index_1, index_2, \ldots, index_n\}$.

Parameter, which are single values, can be entered (for example 5.3, 27.0, 7.38).

Mappings can be chosen with the drop down menu.

### 3.2.2 Implicit data definition

Indexsets, parameter and results can be also specified by a location in a sheet (for example asheet(A1:C7)).

## 3.3 Run the problem

To optimize a problem the run button is selected. A dialog is informing the user on the optimization progress.

# 4 For specialists: defining model building blocks

This chapter assumes prior knowledge within optimization theory and describes the native modeling language.

A text consists of expressions seperated by semicolons. Expressions can be commented. Comments start with # and end with the following line break.

Identifiers start with a letter[1]. A sequence of letters and digits can follow. Numerical values need to be defined with at least one decimal place.

## 4.1 Indexsets

An indexset has a name and consists of one or multiple indices used for naming elements.

**Structure of an indexset expression**
$I : n = \ldots$; (form 1)
or
$I : n = \{i_1, i_2, \ldots, i_n\}$; (form 2)

the identifier $n$ is the name of the indexset and the identifiers $i_1, i_2, \ldots, i_n$ name the indices of the set

Indexsets in form 1 are listed with the data of the derived model items. Form 2 is used in case an indexset is to be defined within the model editor.

## 4.2 Parameter

There is 0-, 1- and 2-dimensional parameter.

[1] $A, \ldots, Z, a, \ldots z$

**Structure of a 0-dimensional parameter expression**
$P : n = \ldots$; (form 1)
or
$P : n = v$; (form 2)

**Structure of a 1-dimensional parameter expression**
$P : n[set_1] = \ldots$; (form 1)

**Structure of a 2-dimensional parameter expression**
$P : n[set_1, set_2] = \ldots$; (form 1)

the identifier $n$ is the name of the parameter, the identifiers $set_1$ and $set_2$ are names of indexsets and $v$ indicates a numerical value

Parameter in form 1 are listed with the data of the derived model items. A parameter in form 2 is a constant value.

## 4.3 Mappings

Mappings index model items derived from the same model building block.

**Structure of a mapping expression**
$J : n[b] = ...$; (form 1)

the identifier $n$ is the name of the mapping and the identifier $b$ is the name of the model building block

Mappings are always listed with the data of the derived model items.

## 4.4 Decision variables

Decision variables come with a type: real, posReal, int, posInt or binary.

**Structure of a k-dimensional variable expression,** $k \in \{0, 1, 2, 3\}$
$V : (type)n[set_1, \ldots, set_k];$
or
$V* : (type)n[set_1, \ldots, set_k];$
or
$V** : (type)n[set_1, \ldots, set_k];$

the identifier $n$ is the name of the variable and $set_1, \ldots, set_k$ are the name of the indexsets

Results are concrete valid values for the decision variables. Starts a variable expression with $V$ :, results can not be displayed. Starts a variable expression with $V*$ : or $V**$ : the results can be configured with the data of the derived model items.

## 4.5 Terms

Terms can be constructed by using braces, arithmetic operators, numerical values, parameter, decision variables and sums.

As braces ( and ) may be used.

The valid arithmetic opertaors are $+$, $-$, $*$ and $\backslash$.

Parameter are referenced by their name and their indices (for example $p[t_3]$).

Decision variables are referenced by their name and their indices (for example $var[t_3]$ or if $m$ is the name of an element of a mapping $m.var_2[t_7]$).

Sums need to have one of the following structures: either $SUM(iINI : t)$ where $I$ is the name of a previously defined indexset and the indentifier $i$ represents a member of this indexset or $SUM(mTYPEM : t)$ where $M$ is the name of a model building block and the identifier $m$ represents a member of this model building block located deeper in the model tree. In both cases $t$ is a term.

Terms multiplying two or more decision variables and dividing by decision variables is excluded.

## 4.6 Constraints

**Structure of a constraints expression**
$n : [FORALL(\ldots)DO]t_1 == t_2;$
or
$n : [FORALL(\ldots)DO]t_1 <= t_2;$
or
$n : [FORALL(\ldots)DO]t_1 >= t_2;$

the identifier $n$ is the name of the constraints and $t_1$ and $t_2$ are valid terms.

The arguments of the $FORALL(\ldots)DO$ can be one or more (in this case seperated by commata) of the following type:

- $iINI$ where $I$ a name of a previously defined indexset and the identifier $i$ represents a member of this indexset
- $mTYPEM$ where $M$ is the name of a model building block and the identifier $m$ represents a member of this model building block located deeper in the model tree.

## 4.7 Objective function

**Structure of an objective expression**
$n : MINt$; (for a minimization problem)
or
$n : MAXt$; (for a maximizition problem)

the identifier $n$ is the name of the objective function and $t$ indicates a term.

# A Appendix

xyzOPT has some special exceptions. If one of the exceptions listed below is thrown, a message dialog appears to inform the user about it.

## A.1 XMLException

Models are saved in xml format. If an xml file to be imported is defective, an XML exception is thrown when it is opened.

## A.2 SyntaxException

If a syntax check is carried out in the model editor and a syntax error is found, a SyntaxException is thrown. This indicates which expression is faulty.

## A.3 RunException

The RunException is thrown if an error is localised while an optimisation calculation is being executed: it is thrown if invalid identifiers were selected during modelling, if the modelling is not linear or if an entry in the input data is syntactically incorrect.

## A.4 SpreadsheetException

xyzOPT supports data import or export to or from an xls file. If an xls file cannot be found or opened, a SpreadsheetException is thrown. A Spreadsheet Exception is also thrown if reading or writing specific values from an xls file is unsuccessful.