Introduction to GAMS Modeling Language

Dr. Davit Stepanyan
M.Sc. Davide Pignotti

Thuenen Institute of Farm Economics, Braunschweig

Approach

- Supplementary GAMS tutorial for the CAPRI training session 2023
- Lectures
- Hands-on exercises
- Discussions
- “Homework” – try to code the models studied during the tutorials from scratch without any help

Structure of the Tutorial (part I)

Wednesday 6th September

1. **GAMS modeling language. General introduction; Introduction to GAMS IDE** (09:00 - 09:45)
2. **MyFarm LP model; Main elements of a GAMS model: variables, parameters, equations** (10:00 - 10:45)
3. **Coding MyFarm LP model in GAMS; Solver output** (11:00 - 11:45)
4. **Solver output; Improving efficiency. Sets, Subsets, Alias, Sum** (12:15 - 13:00)

Thursday 7th September

1. **Introducing Sets and the Sum operator in MyFarm LP; Further useful Gams statements. Prod, Table, Variable-Attributes, Loop** (09:00 - 09:45)
2. **Data exchange with Excel** (10:00 - 10:45)

Objectives

- Upon completion of this tutorial, the participants will be able to:
  - Use the GAMSIDE text editor to modify or code economic simulation models in GAMS
  - Understand the logic behind the GAMS modeling language
  - Differentiate between the main elements of a GAMS model
  - Code simple economic models in GAMS
  - Debug these models
  - Analyze the result files generated by GAMS
  - Exchange data between GAMS and Microsoft Excel

References

- GAMS Documentation (2021). GAMS Development Corporation

What is GAMS?

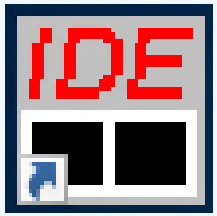
- General Algebraic Modeling System (GAMS) is a high-level modeling system for mathematical programming and optimization.
- GAMS is “a tool for the development, solution, and management of large scale optimization problems”
- Their main distinguishing features are :
  - the use of relational algebra
  - and the ability to provide partial derivatives on multidimensional, very large and sparse structures
- GAMS enables the user to solve/optimize linear as well as non-linear equation systems
  - Optimization problems (maximization/minimization)
  - Fully determined equation systems
  - Combinations
- GAMS consists of a modeling language (along the lines of standard algebra) and solvers to solve or optimize equation systems.

Key Principles of GAMS

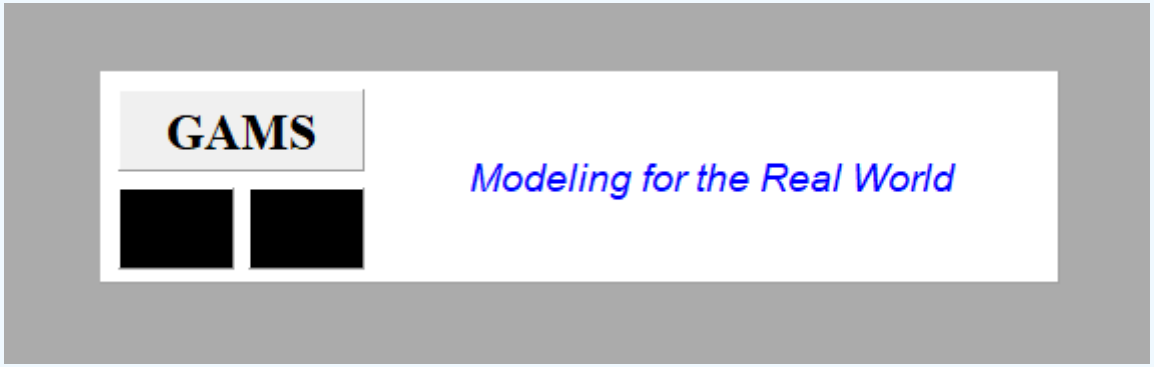
- The problem representation is independent of the solution method.
- The data representation follows the relational data model.
- The problem and data representations are independent of computing platforms.
- The problem and data representations are independent of user interfaces.
- Optimization methods will fail, and systems have to be designed to be fail-safe.

GAMS IDE vs. GAMS Studio

- GAMS IDE
  - Written in Delphi
  - In use for + 20 years
  - Restricted to Windows
- GAMS Studio
  - Written in C++
  - Since 2019
  - Setup similar to IDE
  - Platform-independent



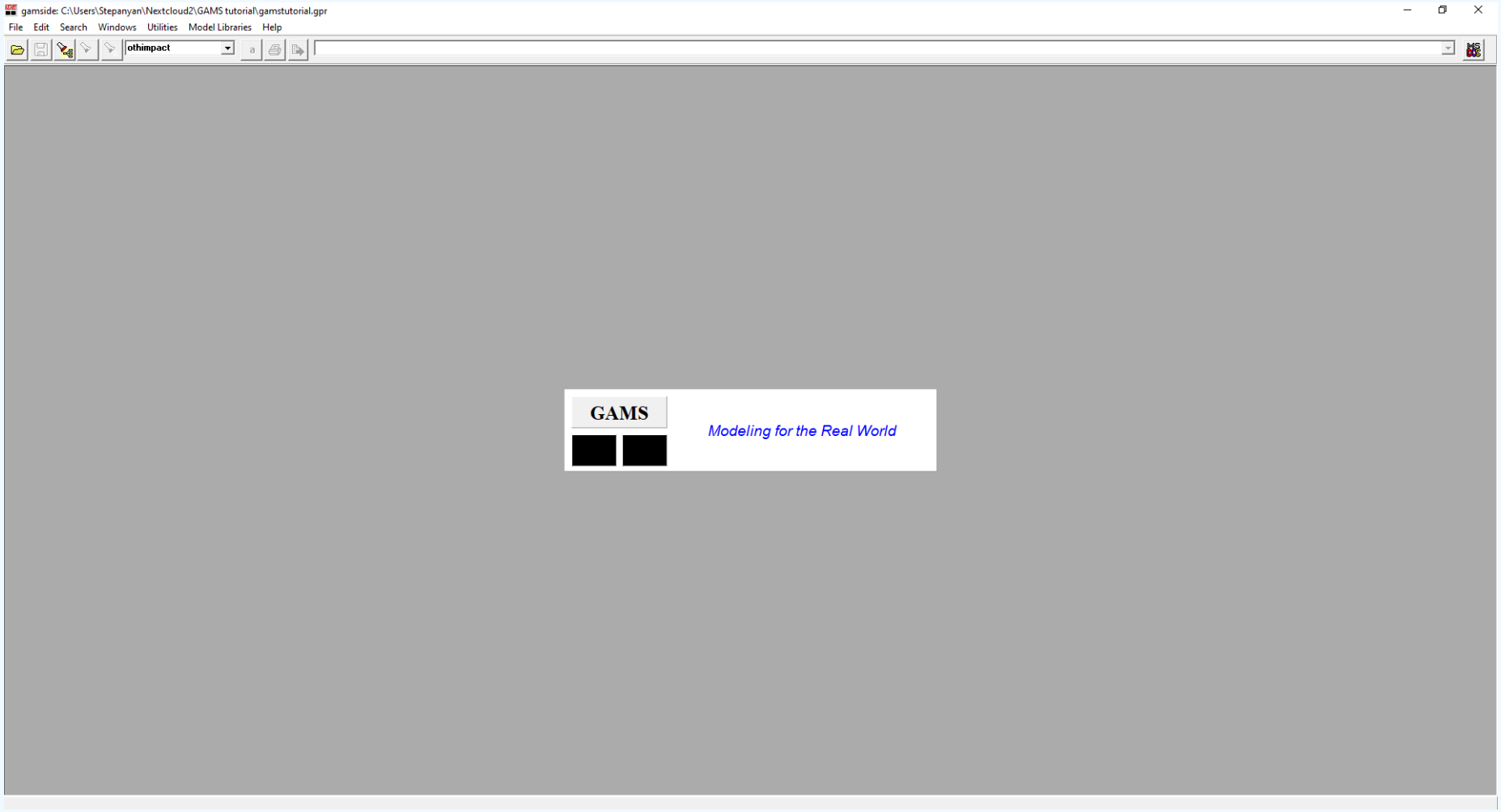
GAMS IDE
(Integrated Development Environment)
Introduction



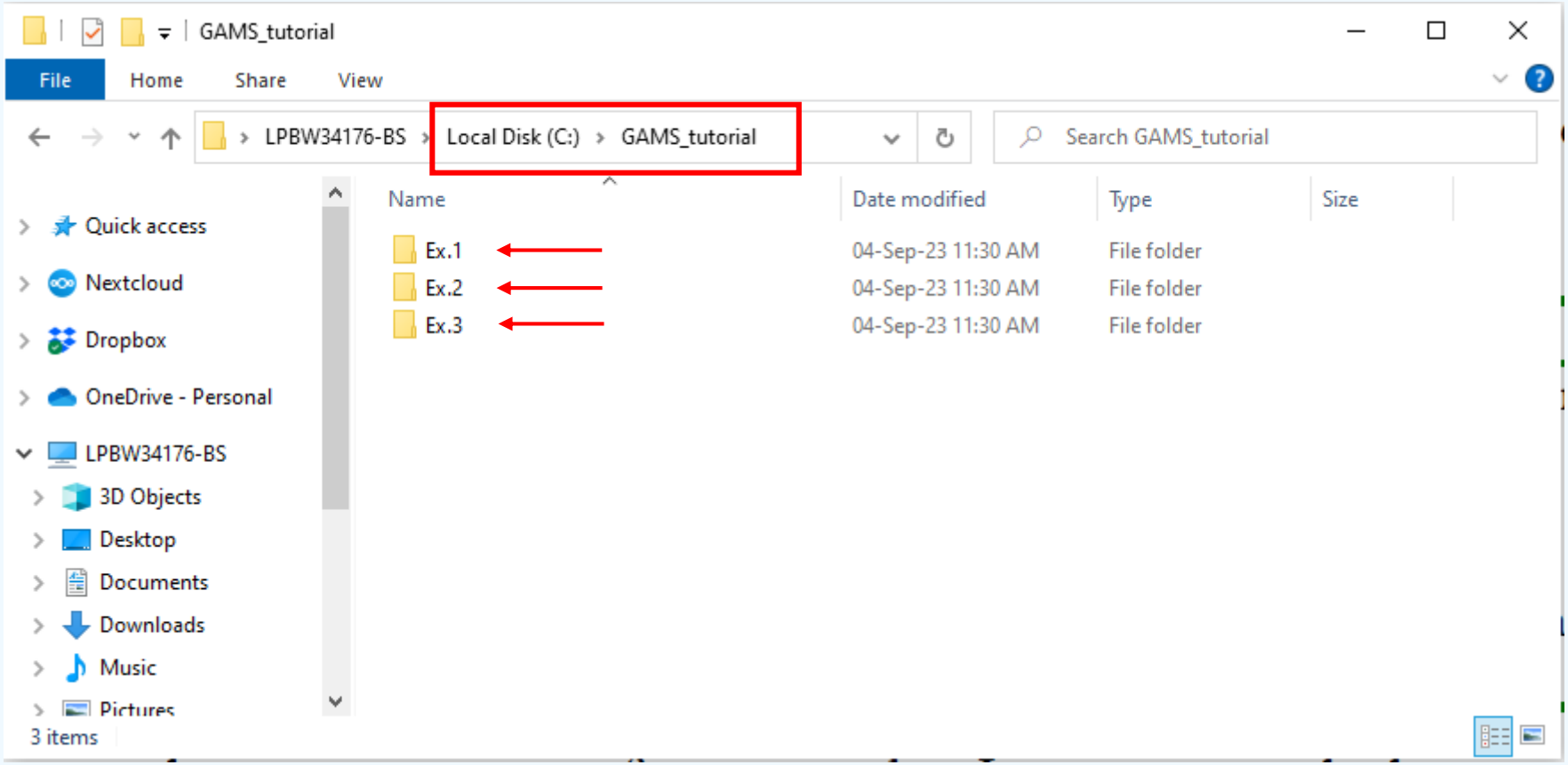
GAMS IDE

- A general text editor with the ability to launch and monitor the compilation/execution of GAMS models
- Progress of a compilation/execution can be monitored in the process window
- The IDE also facilitates the selection of default solvers and manages GAMS parameters on a file by file basis

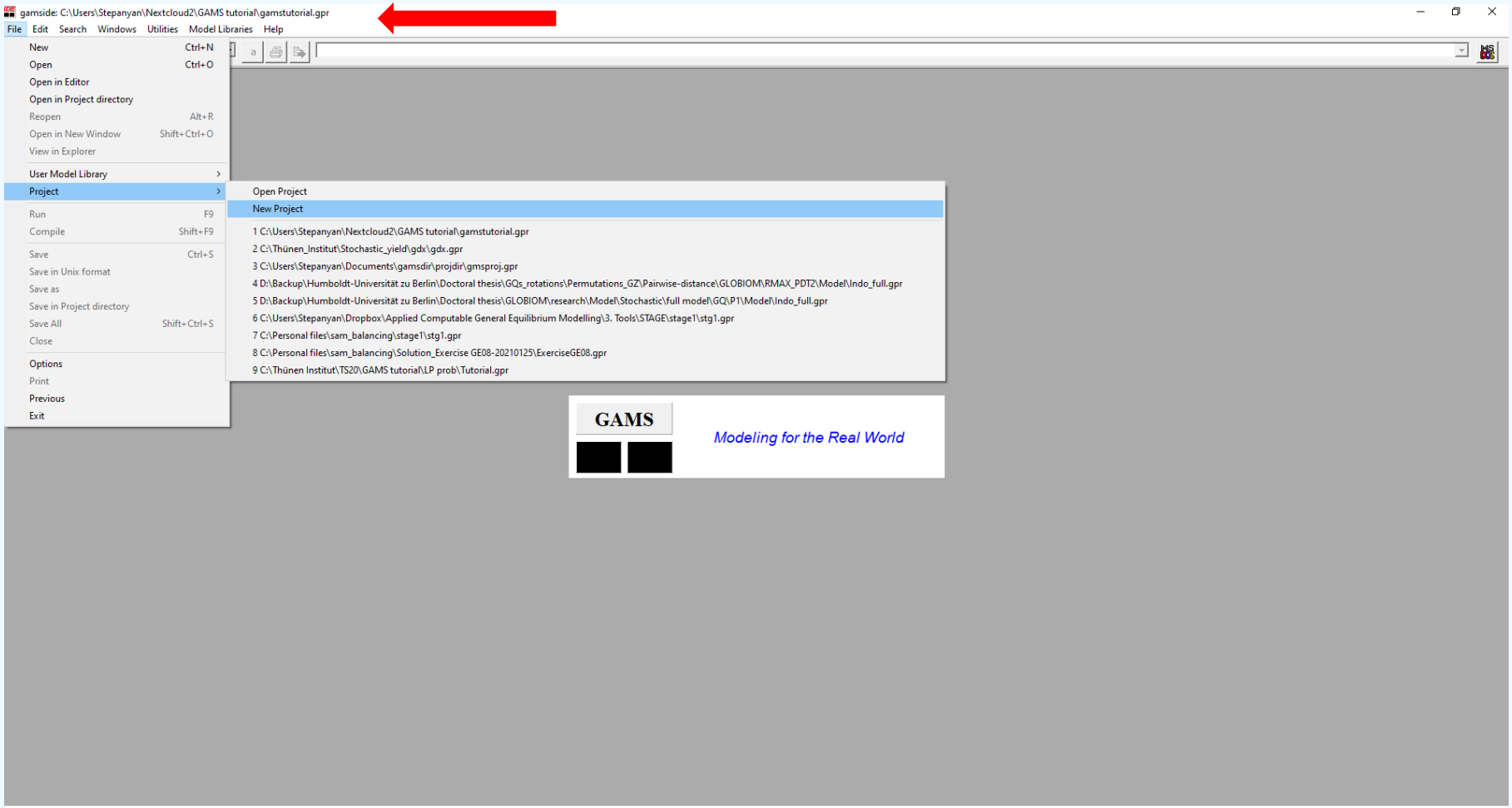
GAMS IDE


gamside: C:\Users\Stepanyan\Nextcloud2\GAMS tutorial\gamstutorial.gpr
File Edit Search Windows Utilities Model Libraries Help
othimpact
GAMS
Modeling for the Real World

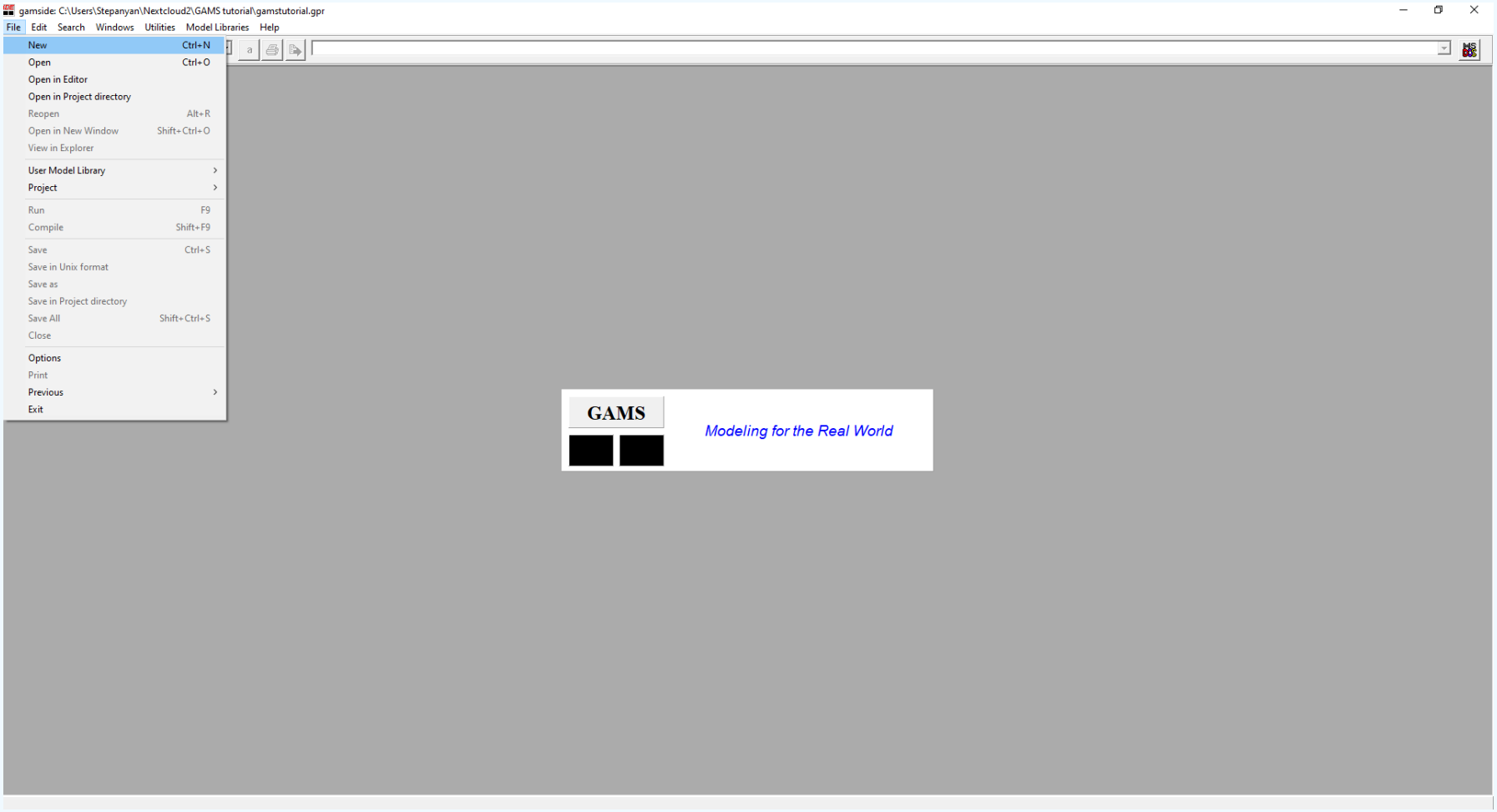
Organization of the Files



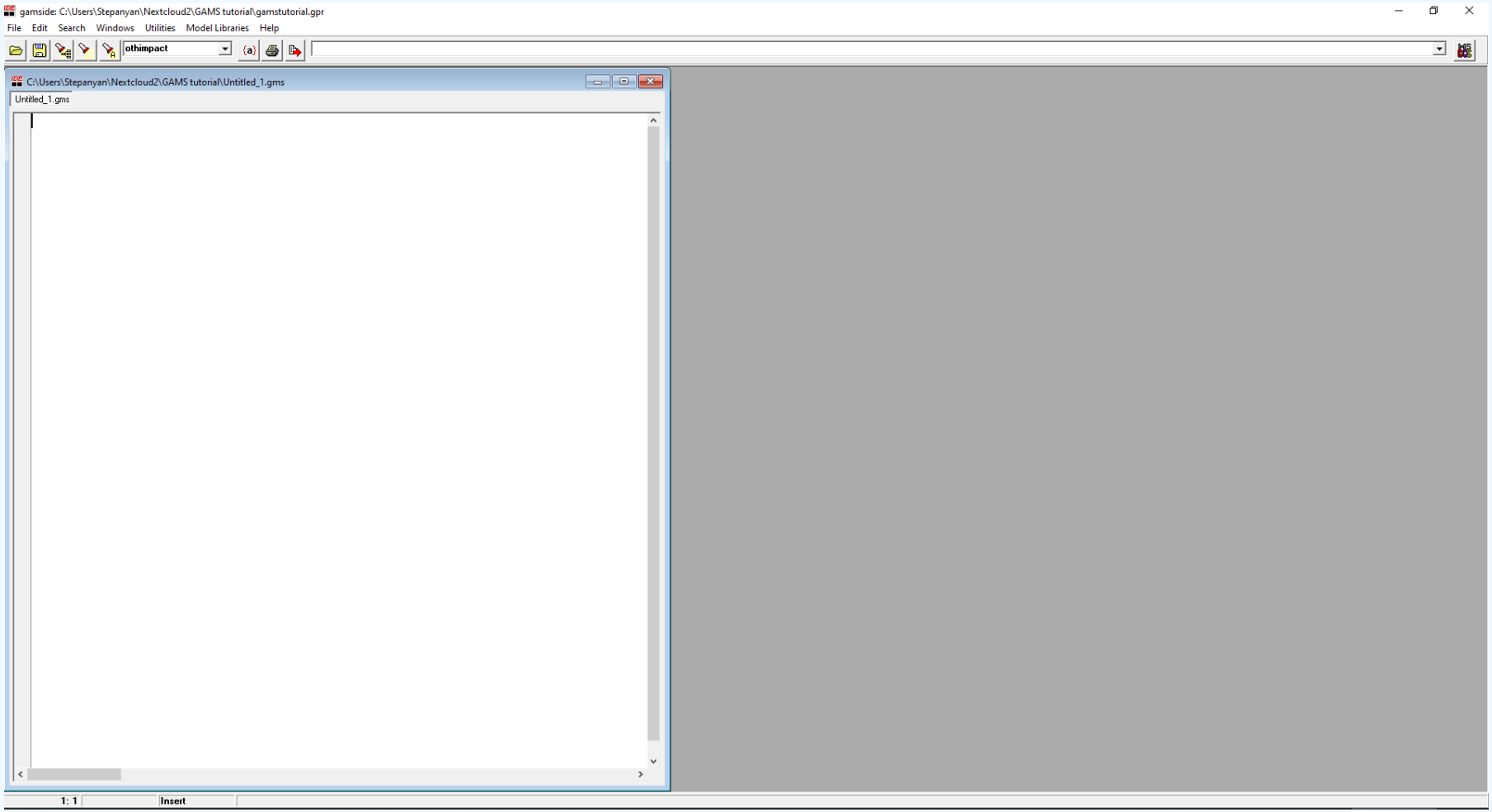
GAMS Project File (.gpr)



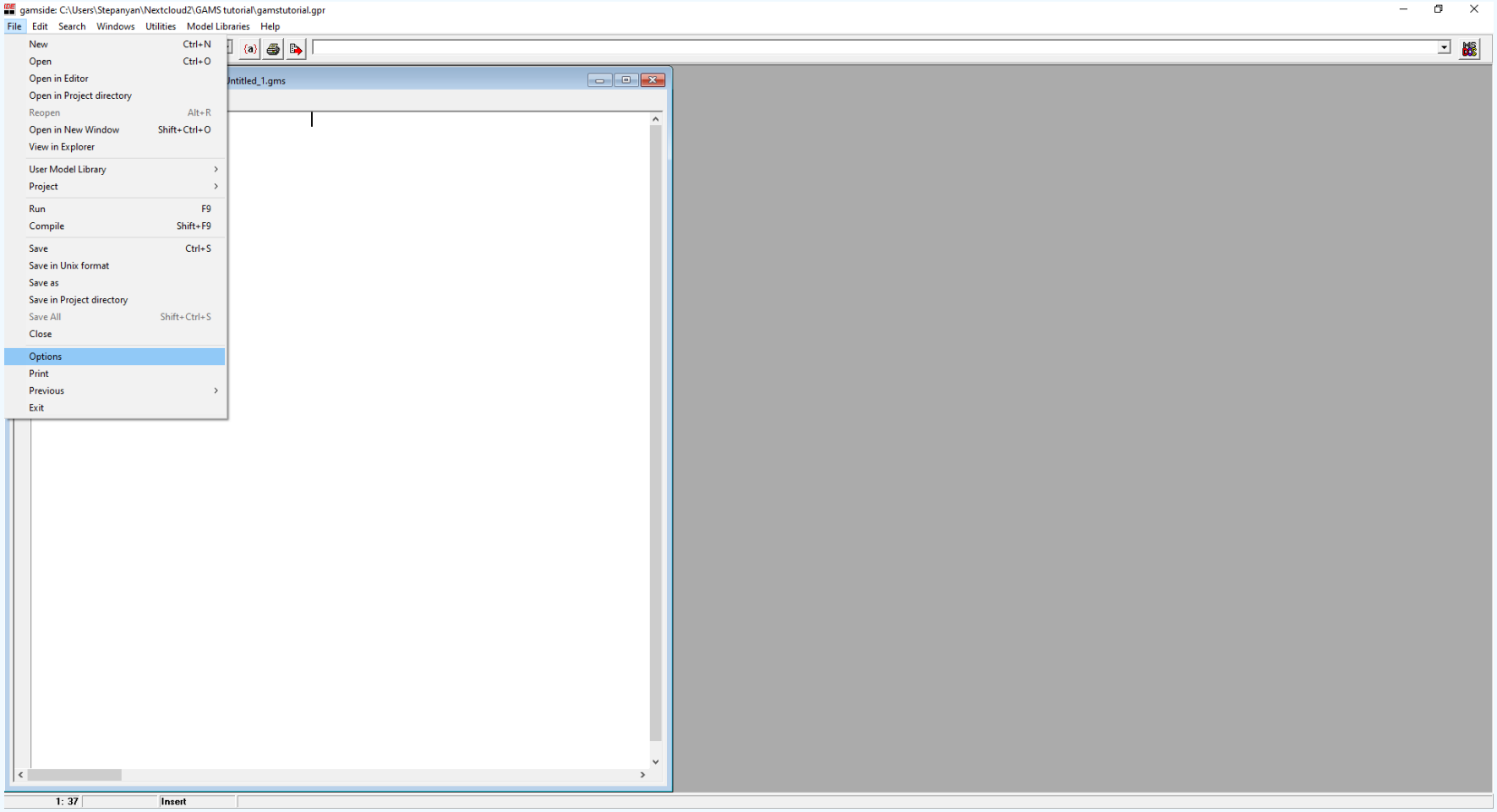
GAMS Files (.gms)



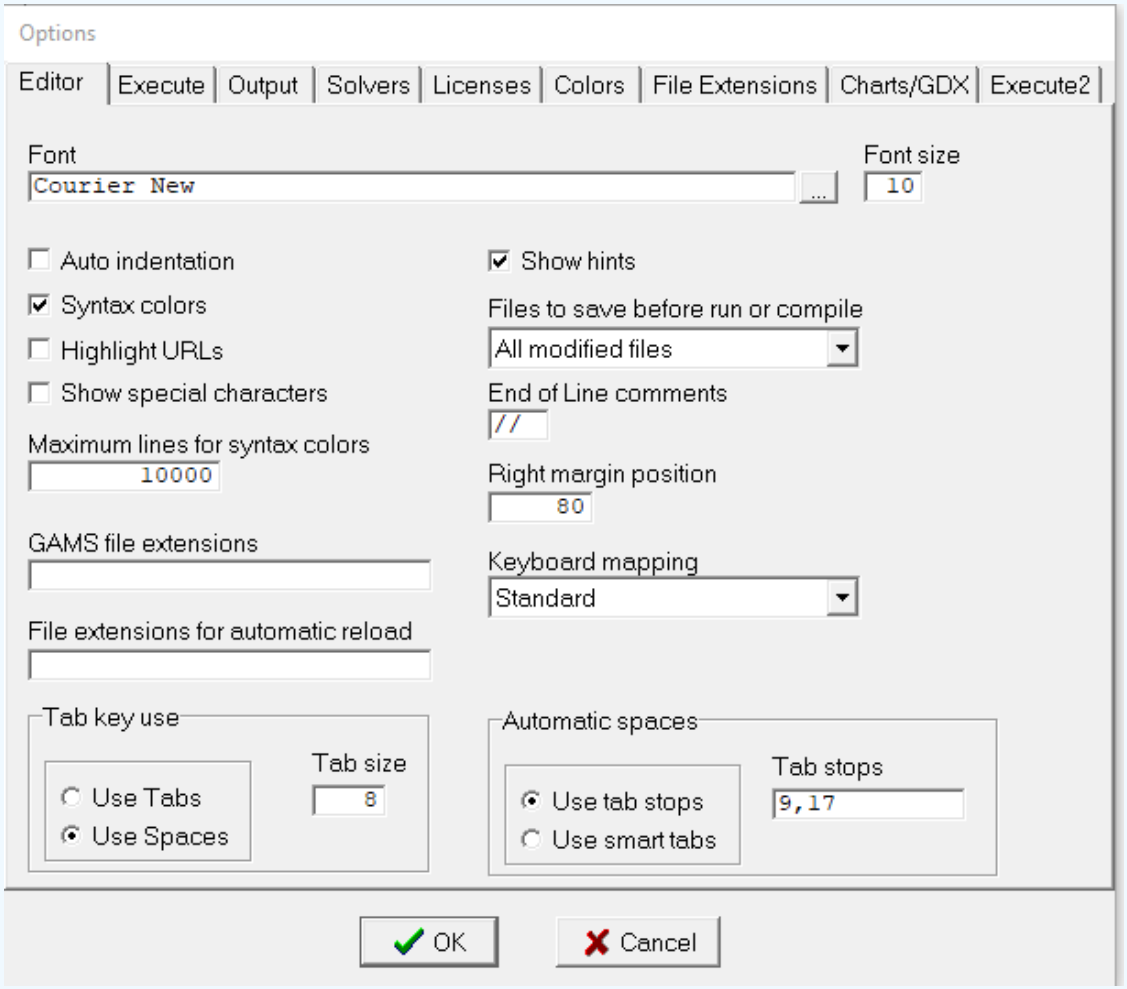
GAMS Files (.gms)



GAMS IDE. Useful Features

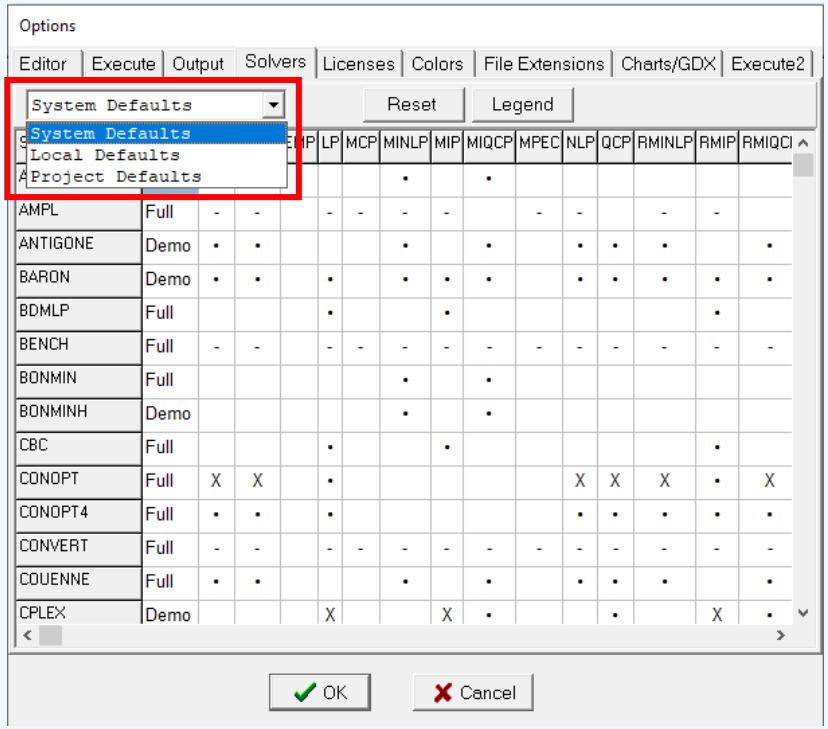


GAMS IDE. Useful Features



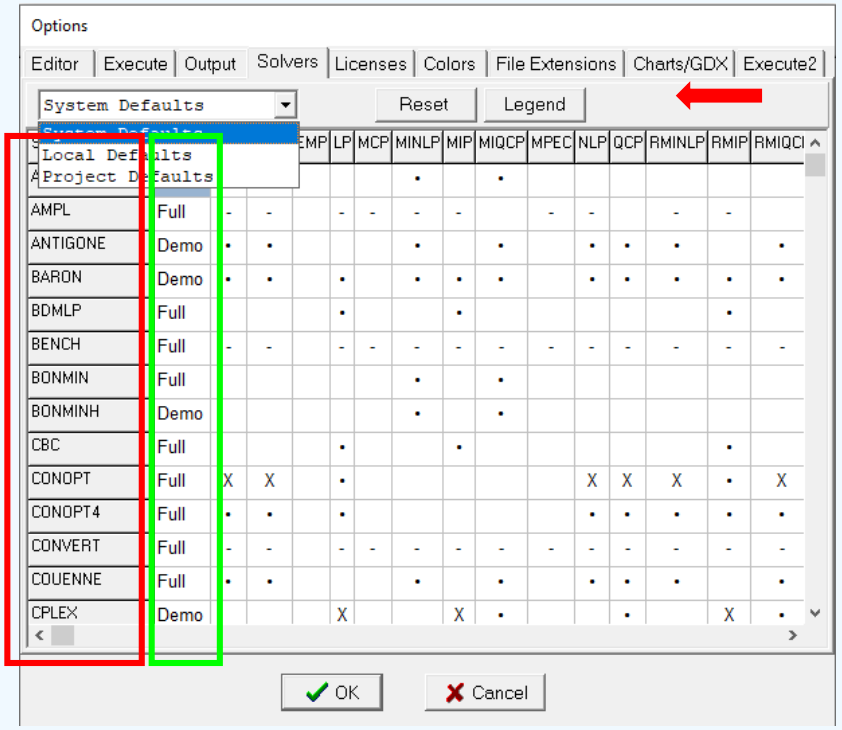
GAMS IDE. Useful Features

- Solvers
  - Matrix of available solvers and model types



GAMS IDE. Useful Features

- Solvers
  - Matrix of available solvers and model types



Available solvers

License status of the solver

Solver Selection Legend

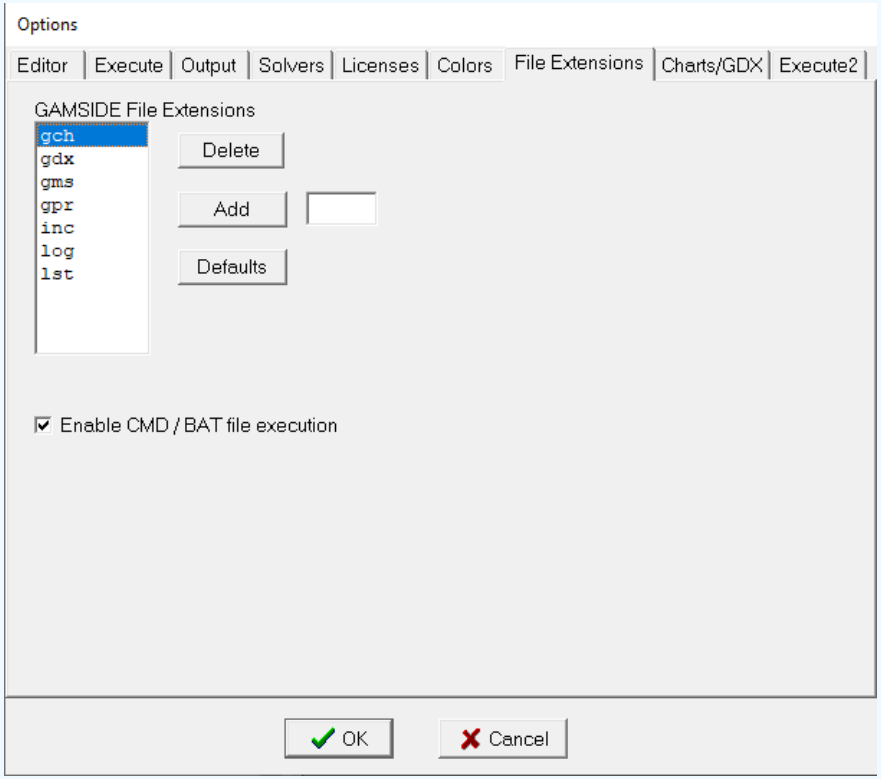
X Current selection

• Available selection

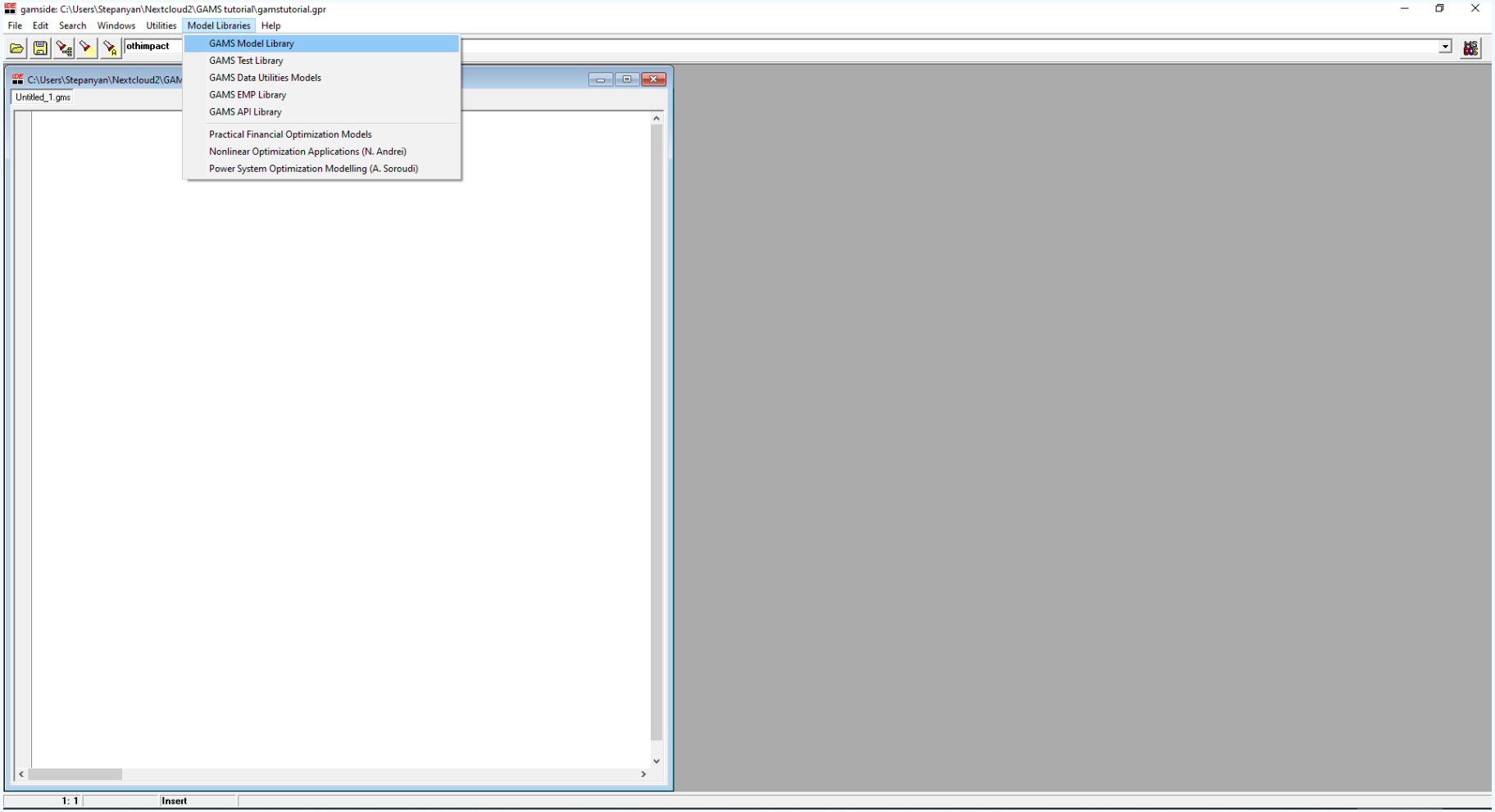
- Not available (Option statement only)

GAMS IDE. Useful Features

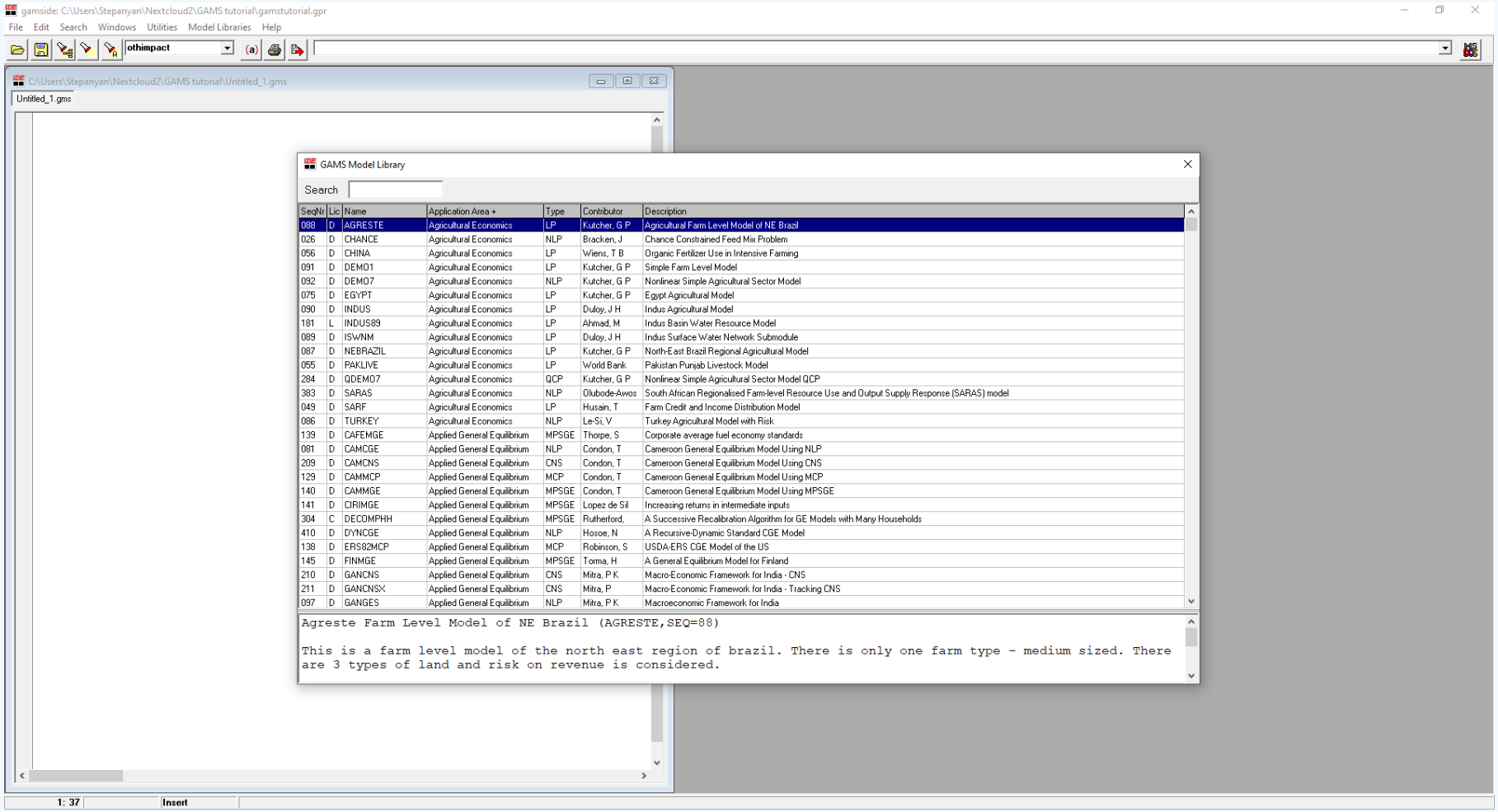
- File Extensions:
  - Select “Defaults”
  - Replace all file extensions with the GAMS standard file extensions ('gms', 'gpr', 'log' and 'lst’).
  - Associate file extensions with the GAMS IDE.



GAMS IDE. Useful Features

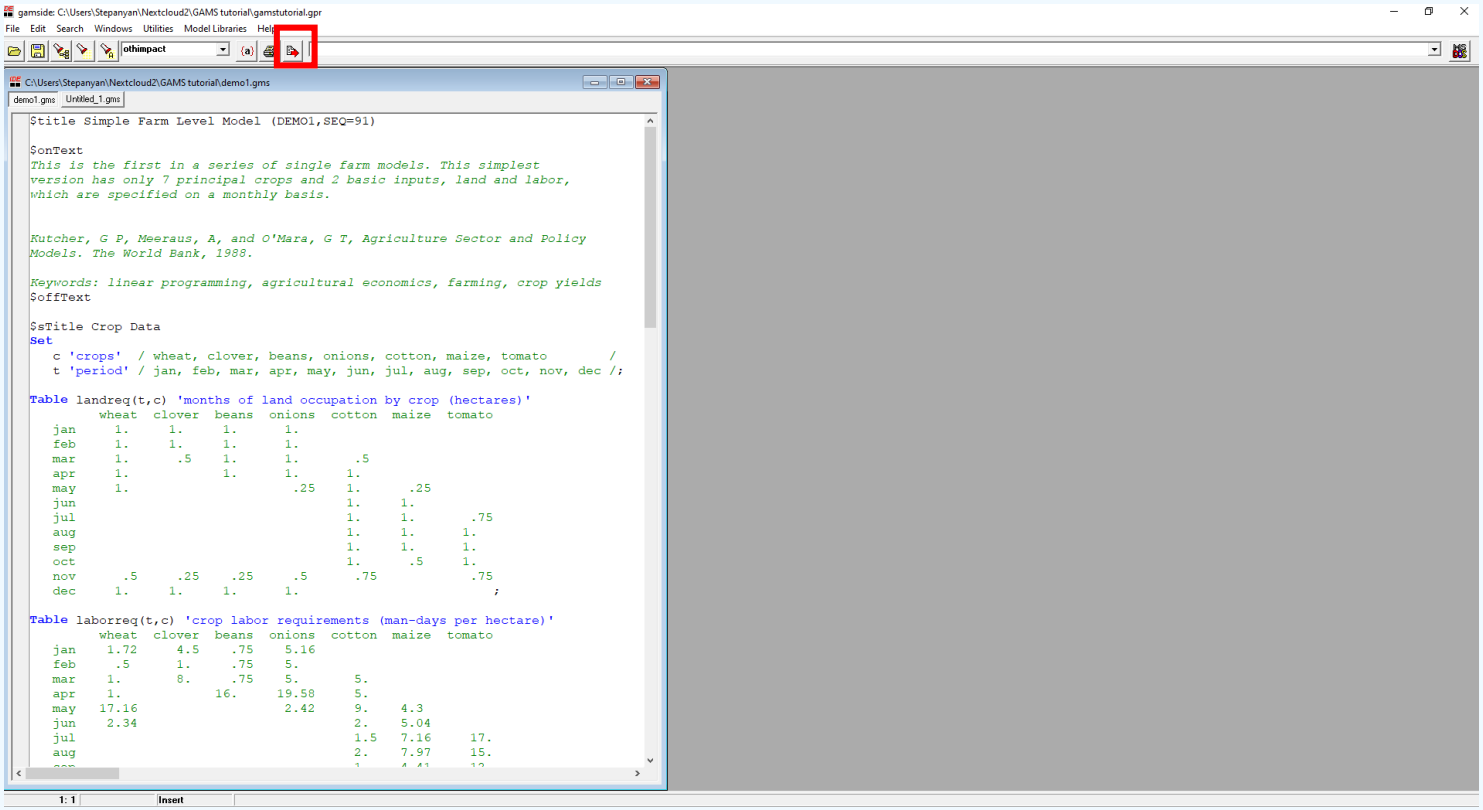


GAMS IDE. Useful Features



Running a Model in GAMS IDE

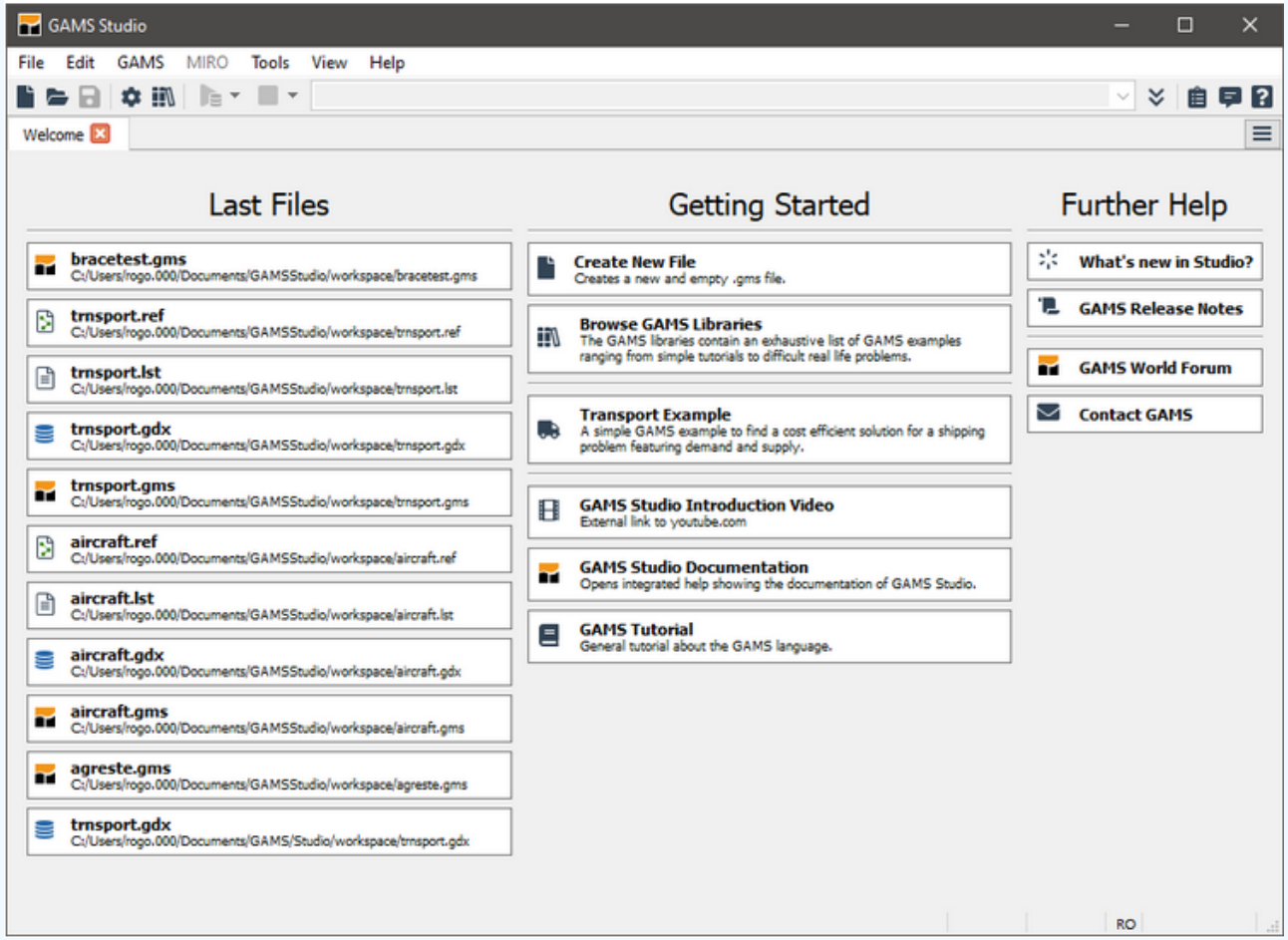
- Or simply press F9



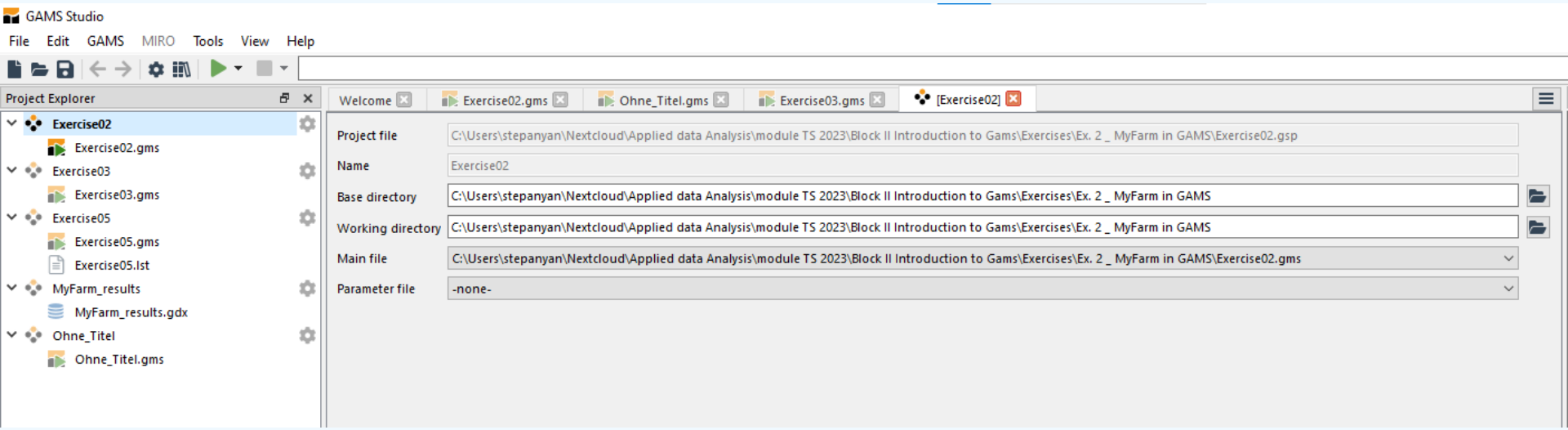
GAMS Studio

Introduction

Welcome Page



Project Handling



Exercise 1. Linear programming model

LP Model

MyFarm

A linear Programming (LP) Problem: “Myfarm” Example

	Wheat	Barley	Rapeseed	Sugarbeet
Gross margin (€/ha)	**253**	**443**	**284**	**516**
Labor requirment (hours/ha)	**25**	**36**	**27**	**87**

- Farms size: 200 ha
- Labor availability: 10000 hours
- X_i: area of land devoted to each crop
- $X_i \geq 0$
- Maximize profit

The Mathematical Model

$$\boldsymbol{Max!}\ \boldsymbol{Z} = 253 * \boldsymbol{X}_{\text{wheat}} + 443 * \boldsymbol{X}_{\text{barley}} + 284 * \boldsymbol{X}_{\text{rapeseed}} + 516 * \boldsymbol{X}_{\text{sugarbeet}}$$

- Subject to:
- $X_{\text{wheat}}\,;\, X_{\text{barley}}\,;\, X_{\text{rapeseed}}\,;\, X_{\text{sugarbeet}} \geq 0$ (non-negativity)
- $X_{\text{wheat}} + X_{\text{barley}} + X_{\text{rapeseed}} + X_{\text{sugarbeet}} \leq 200$ (land)
- $25 * X_{\text{wheat}} + 36 * X_{\text{barley}} + 27 * X_{\text{rapeseed}} + 87 * X_{\text{sugarbeet}} \leq 10000$ (labor)
- Where:

✓ X_{wheat} : land area devoted to wheat production

✓ X_{barley} : land area devoted to barley production

✓ X_{rapeseed} : land area devoted to rapeseed production

✓ $X_{\text{sugarbeet}}$: land area devoted to sugar beet production

MyFarm in Excel

Hands-on exercise

- Open the **Exercise 01-MyFarm in Excel.xlsx** file and solve the LP problem

Questions

- The area devoted to wheat is ___ha. Why?
- Are the land and labour resources fully exploited?
- The max. total gross margin is _____€

The Structure of GAMS Models

Main Elements of a GAMS Model

- 3 essential parts to formulate a GAMS model:
  - Variables
  - Parameters
  - Equations
- To use these in GAMS, 2 steps are required:
  1. Declaration: give it a name and tell GAMS what it is, i.e., parameter, variable, equation, ...
  2. Assignment or definition (a specific value, type, or function)
- Of course, many more statements are available

GAMS Statement: Variables

- Entities whose values are generally unknown until after a model has been solved
- A GAMS variable, like all other identifiers, must be declared before it may be referenced.
- The syntax

```
Variables v_obje;

Positive variables v_actLevlWHEAT, v_actLevlBARLEY, v_actLevlRAPESEED, v_actLevlSUGARBEET;
```

- **Variable(s)** — Keyword for variable definition
- **Positive/Binary** — Keyword can be preceded by modifier:
  - **Positive** — The variable can only contain nonnegative values
  - **Binary** — Only 0 and 1 allowed
- v_actLevlWHEAT, … — List of variable identifiers
- ; — Semicolon ends each GAMS statement

GAMS Statement: Variables. Syntax

```
[var_type] variable[s] var_name [text]
```

Keyword	Description	Default Lower Bound	Default Upper Bound
`free` (default)	No bounds on variable. Both bounds may be changed from the default values by the user.	`-inf`	`+inf`
`positive` or `nonnegative`	No negative values are allowed for variable. The user may change both bounds from the default value.	0	`+inf`
`negative`	No positive values are allowed for variables. The user may change both bounds from the default value.	`-inf`	0
`binary`	Discrete variable that can only take values of 0 or 1. For details see section **Types of Discrete Variables**. In **relaxed Model types** the integrality requirement is relaxed.	0	1
`integer`	Discrete variable that can only take integer values between the bounds. The user may change both bounds from the default value. The default upper bound inside GAMS is `+inf` but when the variable is passed on to the solver, the option or command line parameter **IntVarUp** decides what upper bound (by default 100) is passed on to the solver in case GAMS has upper bound +inf. In **relaxed Model types** the integrality requirement is relaxed.	0	`+inf`
`sos1`	A set of variables, such that at most one variable within a group may have a non-zero value. For details see section **Types of Discrete Variables**.	0	`+inf`
`sos2`	A set of variables, such that at most two variables within a group may have non-zero values and the two non-zero values are adjacent. For details see section **Types of Discrete Variables**.	0	`+inf`
`semicont`	Semi-continuous, must be zero or above a given minimum level. For details see section **Types of Discrete Variables**.	1	`+inf`
`semiint`	Semi-integer, must be zero or above a given minimum level and integer. For details see section **Types of Discrete Variables**. The default upper bound inside GAMS is `+inf` but when the variable is passed on to the solver, the option or command line parameter **IntVarUp** decides what upper bound (by default 100) is passed on to the solver in case GAMS has upper bound +inf. In **relaxed Model types** the integrality requirement is relaxed.	1	`+inf`

GAMS Statement: Variables

- Good modeling practice:
  - Use expressions which are short (one word) but are still telling
    - Not VAR1, A, B
  - Add explanatory text to further clarify the meaning:

```
Variables
v_obje              objective function value
;
Positive variables
v_actLevlWHEAT      land area wheat
v_actLevlBARLEY     land area barley
v_actLevlRAPESEED   land area rapeseed
v_actLevlSUGARBEET land area sugar beet
;
```

GAMS Statement: Parameters

- Are constants and contain exogenously given values
- Are not modified during the solution process

```
Parameters
p_uvag_wheat      Gross margin of wheat              /253/,
p_uvag_barley     Gross margin of corn               /443/,
p_uvag_rapeseed   Gross margin of rapeseed           /284/,
p_uvag_sugarbeet Gross margin of sugar beet         /516/,
;
```

- Where
  - `Parameters` — Keyword for parameter definition
  - `p_uvag_wheat` — Identifier of the parameter
  - `/253/` — Initial assignment of a value (otherwise: 0)
  - `,` — To define more parameters in one statement, separate them by commas or breaks
  - `;` — Every GAMS statement is concluded by a semicolon

GAMS Statement: Parameters

- Good modeling practice: Separate declaration and assignment

```
Parameters
p_uvag_wheat     Gross margin of wheat
p_uvag_barley    Gross margin of barley
;
p_uvag_wheat  =  253;
p_uvag_barley = 443;
```

- Assignment may contain arithmetic operations, e.g.

Operation	Symbol	Order of Precedence
Exponentiation	**	1
Multiplication	*	2
Division	/	2
Addition	+	3
Subtraction	-	3

GAMS Statement: Parameters

- Values of parameters can be “overwritten”:

```
Parameters
p_uvag_wheat     Gross margin of wheat
gm_barley    Gross margin of barley
;
p_uvag_wheat  = 253;
p_uvag_barley = 443;

p_uvag_wheat = 378;
```

GAMS Statement: Parameters

- To check the value of a parameter or variable use the display statement:

```
DISPLAY p_uvag_wheat;

or

Display p_uvag_wheat;

or

display p_uvag_wheat;

or

displAy p_uvag_wheat;
```

GAMS Statement: Equations

- In GAMS, each equation consists of 2 separate statements:
  - Declaration (declare the equations existence)
  - Definition (the equation itself, its algebraic form)
- Declaration:

```
Equations e_land, e_labour, obje;
```

- Where
  - `Equations` — Keyword for equation declaration
  - `e_land,… obje` — List of equations to be declared
  - `;` — End GAMS statement

GAMS Statement: Equations

- Definition
  - Equation name followed by two dots (..)
  - Then the algebraic form of the equation
  - In equations, the relational operators (= ≤ ≥) must be written as:

=E=	Equality: right-hand side must equal left-hand side
=G=	Greater than: left-hand side must be greater than or equal to right-hand side
=L=	Less than: left-hand side must be less than or equal to right-hand side

GAMS Statement: Equations

- Definition
  - Equation definitions for the MyFarm-exercise:

```
obje ..      v_obje =E= p_uvag_wheat * v_actLevlWHEAT + p_uvag_barley *
v_actLevlBARLEY + p_uvag_rapeseed * v_actLevlRAPESEED + p_uvag_sugarbeet *
v_actLevlSUGARBEET;

e_land ..    v_actLevlWHEAT + v_actLevlBARLEY + v_actLevlRAPESEED +
v_actLevlSUGARBEET =L= 200;

e_labour ..    p_lab_wheat * v_actLevlWHEAT + p_lab_barley * v_actLevlBARLEY +
p_lab_rapeseed * v_actLevlRAPESEED + p_lab_sugarbeet * v_actLevlSUGARBEET =L= 10000;
```

GAMS Statement: Equations

- Definition
  - Note: The general form of these statements is

```
equationname.. algebra1 relation algebra2;
```

- Where

`equationname`	The identifier of the equation as declared
`..`	Separator between name and equation
`algebra1,algebra2`	Some algebraic expressions containing parameters and at least one endogenous variable
`relation`	One of the following =E= or =L= or =G=
`;`	End of the statement

GAMS Statement: Model

- Once all the model structural elements have been defined, the model has to be defined by a `Model` statement to identify the equations that belong to the model.

```
Model myfarm /e_land, e_labour, obje/;

Or

Model myfarm /all/;
```

- Where
  - `Model` Keyword for model definition
  - `Myfarm` Identifier for this model
  - `/e_land, …/` List of equations that belong to this model
  - `;` Ends the statement
- The keyword `all` includes all previously defined equations in the model.

GAMS Statement: Solve

- The `Solve` statement causes GAMS to use a solver to optimize/solve the model

```
Solve myfarm using lp maximizing v_obje;
```

- Where
  - `solve` Keyword for the solve statement
  - `myfarm` Name of the model to be solved
  - `using lp` Declares type of solver to be used (lp = linear programming)
  - `maximizing` Declares the direction of the optimization (alternative: minimizing)
  - `v_obje` Target variable (has to be defined before and has to occur in the model, must not be restricted)
  - `;` Ends the statement

Good Modelling Practices

- Enhance the readability of the model (to others/ to you after some time of absence) by:
  - Ordering: e.g. first define all parameters, variables,…then assign them
  - Giving telling names to model-entries (“p_uvag_wheat” instead of “par_1”)
  - Defining all entries using explanatory text
  - Specify units of all entries (e.g. ha, 1000 USD,…)
  - Use comments wherever useful:

```
* starts a comment-line

$ontext
starts a comment which can stretch over several lines and needs to
be ended by
$offtext
```

Exercise 2. Linear programming model in GAMS

Coding the MyFarm Model

MyFarm Model

- The problem

  Farmer wants to maximize his profit given 100 ha of land and 500 hours of labor. He/she has the option of cultivating wheat, barley rapeseed and/or sugarbeet. How much of each crop should be plated in order to maximize the profit.

	Wheat	Barley	Rapeseed	Sugarbeet
Gross margin (€/ha)	**253**	**443**	**284**	**516**
Labor requirment (hours/ha)	**25**	**36**	**27**	**87**

Questions

- The value of an additional unit of land is _____€
- If we include 1 ha of wheat in the cropping mix, the tot. gross margin will increase/decrease by _____€

Solver Output: the Listing File

The Listing File ".lst"

- The output file generated from a GAMS run is called listing file
- The listing file contains (in standard settings)
  - Echo print
  - Error messages
  - Equation listing
  - Column listing (variables)
  - Model statistics
  - Solution report
  - SolEQU, SolVAR:
    - Solution values for equations and variables

The Listing File “Echo print”

- Is always the first part of the output file
- It is a listing of the input with added line numbers

```
22  *$title MyFarm
23  Variables
24     v_obje  objective function value
25  ;
26
27  Positive variables
28     v_actLevlWHEAT        land area wheat
29     v_actLevlBARLEY       land area barley
30     v_actLevlRAPESEED     land area rapeseed
31     v_actLevlSUGARBEET    land area sugarbeet
32  ;
33
34  Parameters
35     p_uvag_wheat              Gross margin of wheat               /253/ ,
36     p_uvag_barley             Gross margin of barley              /443/ ,
37     p_uvag_rapeseed           Gross margin of rapeseed            /284/ ,
38     p_uvag_sugarbeet          Gross margin of sugarbeet           /516/ ,
39
40     p_lab_wheat               Required labor hours of wheat       /25/  ,
41     p_lab_barley              Required labor hours of barley      /36/  ,
42     p_lab_rapeseed            Required labor hours of rapeseed    /27/  ,
43     p_lab_sugarbeet           Required labor hours of sugarbeet   /87/
44  ;
45
46
47  Equations
48  *Declaration
49
50    e_land   land constraint
51    e_labour labour constraint
52    obje     objective function
53  ;
54
55  *Defintion
56  obje ..       v_obje =E= p_uvag_wheat * v_actLevlWHEAT + p_uvag_barley * v
    _actLevlBARLEY + p_uvag_rapeseed * v_actLevlRAPESEED + p_uvag_sugarbeet *
    v_actLevlSUGARBEET;
57
58  e_land ..     v_actLevlWHEAT + v_actLevlBARLEY + v_actLevlRAPESEED + v_act
    LevlSUGARBEET =L= 200;
59
60  e_labour ..   p_lab_wheat * v_actLevlWHEAT + p_lab_barley * v_actLevlBARLE
    Y + p_lab_rapeseed * v_actLevlRAPESEED + p_lab_sugarbeet * v_actLevlSUGARB
    EET =L= 10000;
61
62
63
64  Model myfarm /all/;
65
66  Solve myfarm using lp maximizing v_obje;
```

The Listing File “Error messages”

- All errors are marked by 4 stars (****)
  - Search for **** to find errors in the listing file
- Two types of error messages:
  1. Compilation errors (syntax/consistency mistakes)
     - `$errornumber` is placed below the exact position in the line where the error occurred
     - `errornumber` is referenced by an error listing that describes this error
  2. Execution errors (illegal arithmetic operations)
     - Errors after compilation finished, i.e., during model generation and solving

The Listing File “Error messages” Examples

- Compilation error
  - A dollar symbol and error number are printed below the offending symbol on a separate line that begins with four asterisks

```
  55  *Defintion
  56  obje ..       v_obje =E= p_uvag_whea * v_actLevlWHEAT + p_uvag_barley * v_
****                                        $140
      actLevlBARLEY + p_uvag_rapeseed * v_actLevlRAPESEED + p_uvag_sugarbeet * v
      _actLevlSUGARBEET;
**** 140  Unknown symbol
```

- Execution error
  - Occurs after compilation has finished

```
58  e_land ..     v_actLevlWHEAT + v_actLevlBARLEY + v_actLevlRAPESEED + v_act
    LevlSUGARBEET =L= 200/0;
```

```
**** Exec Error at line 58: division by zero (0)
```

The Listing File “Equation listing”

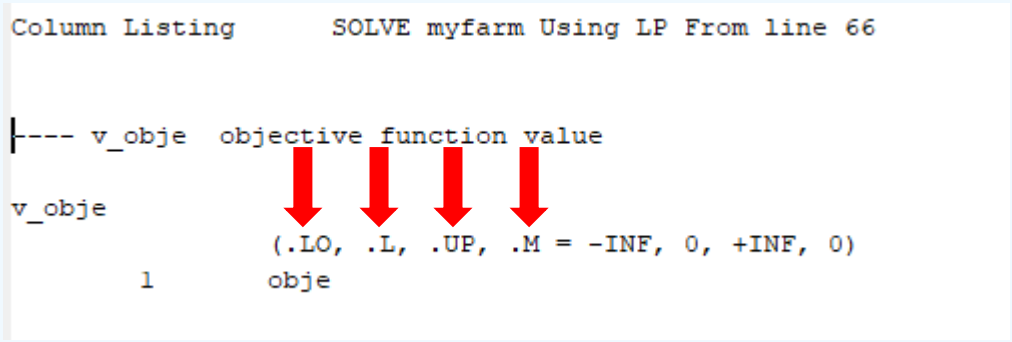
- Is the first part of the output generated by a solve statement
- By default, the first three equations in every block are listed
  - This can be modified with the option `limrow`

```
Equation Listing    SOLVE myfarm Using LP From line 66


---- e_land  =L=  land constraint

e_land..  v_actLevlWHEAT + v_actLevlBARLEY + v_actLevlRAPESEED

      + v_actLevlSUGARBEET =L= 200 ; (LHS = 0)


---- e_labour  =L=  labour constraint

e_labour..  25*v_actLevlWHEAT + 36*v_actLevlBARLEY + 27*v_actLevlRAPESEED

      + 87*v_actLevlSUGARBEET =L= 10000 ; (LHS = 0)


---- obje  =E=  objective function

obje..  v_obje - 253*v_actLevlWHEAT - 443*v_actLevlBARLEY

      - 284*v_actLevlRAPESEED - 516*v_actLevlSUGARBEET =E= 0 ; (LHS = 0)
```

- For equalities: LHS should be equal to RHS, typically 0.

The Listing File “Column listing”

- The column listing or variable listing is the next part of the output

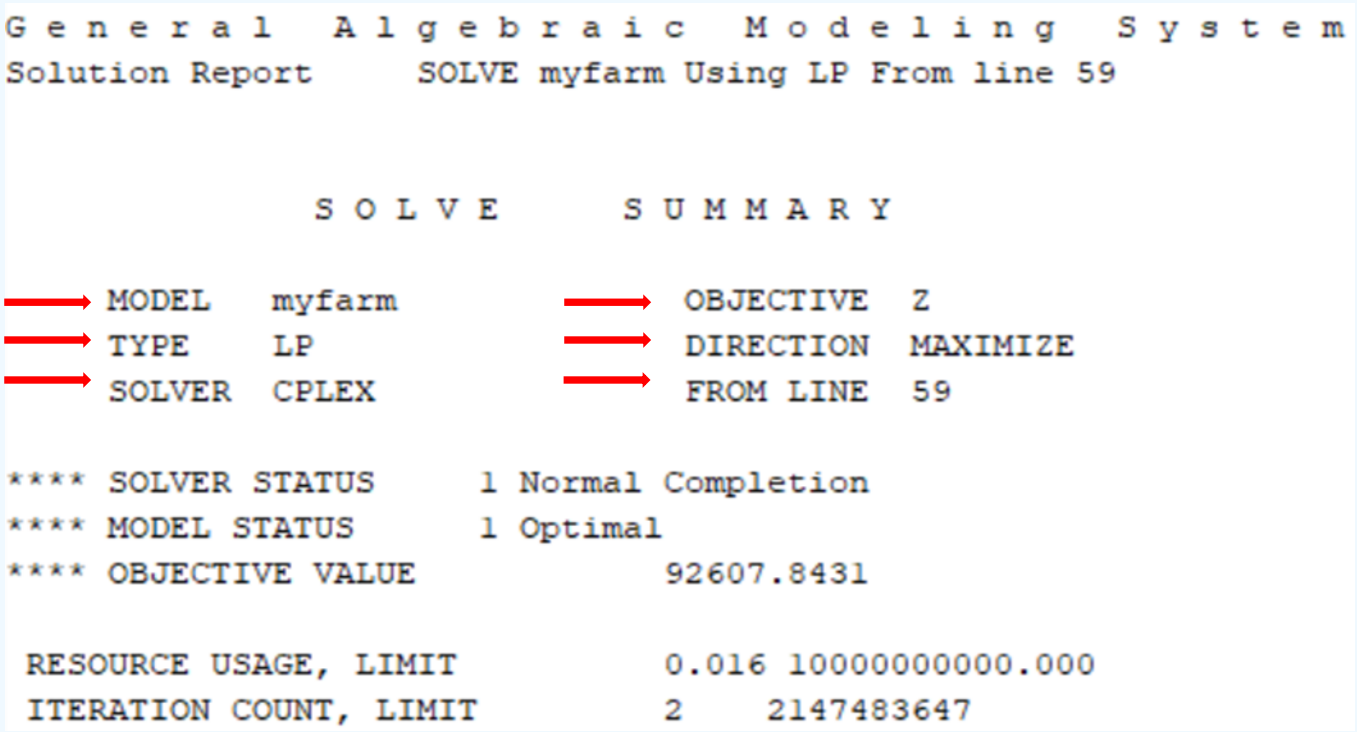


The Listing File “Model statistics”

```
Model Statistics    SOLVE myfarm Using LP From line 59


MODEL STATISTICS

BLOCKS OF EQUATIONS           3     SINGLE EQUATIONS            3
BLOCKS OF VARIABLES           5     SINGLE VARIABLES            5
NON ZERO ELEMENTS            13


GENERATION TIME      =        0.015 SECONDS      3 MB  39.2.1 98a2c774 WEX-WEI
```

The Listing File “Solution report”

- It is marked with the title Solution Report and includes the solve summary, the solver report, the solution listing, and the report summary



```
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Solution Report     SOLVE myfarm Using LP From line 59


               S O L V E      S U M M A R Y

     MODEL   myfarm          OBJECTIVE  Z
     TYPE    LP              DIRECTION  MAXIMIZE
     SOLVER  CPLEX           FROM LINE  59

**** SOLVER STATUS     1 Normal Completion
**** MODEL STATUS      1 Optimal
**** OBJECTIVE VALUE              92607.8431

 RESOURCE USAGE, LIMIT          0.016 10000000000.000
 ITERATION COUNT, LIMIT         2    2147483647
```

The Listing File “The Solution Listing”

- The solution listing is a row-by-row then column-by-column listing of the solutions returned to GAMS by the solver program
- Each individual equation (SolEQU) and variable (SolVAR) is listed with four pieces of information
  - May be suppressed by

```
option solprint = off ;
```

- SolEQU:

Shadow price

	LOWER	LEVEL	UPPER	MARGINAL
---- EQU e_land	-INF	200.000	200.000	391.471
---- EQU e_labour	-INF	10000.000	10000.000	1.431
---- EQU obje	.	.	.	1.000

The Listing File “The Solution Listing”

- SolVAR:

Reduced cost

	LOWER	LEVEL	UPPER	MARGINAL
---- VAR v_obje	-INF	92607.843	+INF	.
---- VAR v_actLevl~	.	.	+INF	-174.255
---- VAR v_actLevl~	.	145.098	+INF	.
---- VAR v_actLevl~	.	.	+INF	-146.118
---- VAR v_actLevl~	.	54.902	+INF	.

The Listing File “Report Summary”

```
**** REPORT SUMMARY :          0     NONOPT
                               0 INFEASIBLE
                               0  UNBOUNDED
```

Improving the Efficiency of Coding in GAMS

Introducing Sets in the GAMS model

The Mathematical Model

$$\boldsymbol{Max!}\ \boldsymbol{Z} = 253 * \boldsymbol{X}_{\text{wheat}} + 443 * \boldsymbol{X}_{\text{barley}} + 284 * \boldsymbol{X}_{\text{rapeseed}} + 516 * \boldsymbol{X}_{\text{sugarbeet}}$$

- Subject to:
- $X_{\text{wheat}}\,; X_{\text{barley}}\,; X_{\text{rapeseed}}\,; X_{\text{sugarbeet}} \geq 0$ (non-negativity)
- $X_{\text{wheat}} + X_{\text{barley}} + X_{\text{rapeseed}} + X_{\text{sugarbeet}} \leq 200$ (land)
- $25 * X_{\text{wheat}} + 36 * X_{\text{barley}} + 27 * X_{\text{rapeseed}} + 87 * X_{\text{sugarbeet}} \leq 10000$ (labor)
- Where:

✓ X_{wheat} : land area devoted to wheat production

✓ X_{barley} : land area devoted to barley production

✓ X_{rapeseed} : land area devoted to rapeseed production

✓ $X_{\text{sugarbeet}}$: land area devoted to sugar beet production

An Alternative Formulation

Max $\sum_j c_j X_j$

s.t. $\sum_j a_{ij} X_j \leq b_i$

$X_j \geq 0$

Where

- j = {wheat, barley, rapeseed, sugarbeet}
- i = {land, labor}
- X_j = $\{X_{wheat}, X_{barley}, X_{rapeseed}, X_{sugarbeet}\}$
- C_j = {253, 443, 284, 516}
- a_{ij} =$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 25 & 36 & 27 & 87 \end{pmatrix}$
- b_i = {200, 10000}

GAMS Statement: Sets

- If we have several parameters or variables of the same type, we can use sets as indices for these parameters or variables

```
SET COLS /wheat, barley, rapeseed, sugarbeet/;
```

- Where

`Set(s)`	Keyword for set definition
`COLS`	Identifier of the set
`/wheat, …/`	List of set elements

- Use of a set in a parameter definition:

```
Parameter p_uvag(crops)    /wheat      253,
                            barley     443,
                            rapeseed   284,
                            sugarbeet 516/;
```

GAMS Statement: Sets

- Assigning all the parameters of the set, overwriting it or referring to it:

```
p_uvag(crops) = 90;
```

- Referring to a single element of a set or overwriting it :

```
p_uvag(“wheat”) = 90;
```

GAMS Statement: Sets

- The format for the set declaration and element definition statement is:

```
SET         setname                 optional explanatory text
      /first set-element name       optional explanatory text
      second set-element name       optional explanatory text
      …
      /;
```

  - The word **set** or **sets** can be used.
  - Multiple sets can be stacked with the set or sets keyword only used once.
  - When multiple sets are defined in one set statement a “;” is entered after all set definitions only:

```
SETs      j   /x1,x2,x3/
          i   /r1
               r2/;
```

  - Set elements are separated by commas or by an end-of-line.

Sequences as Set Elements

- The asterisk '*' plays a special role in set definitions.
  - Used to define sequence of elements for a set

```
Set t "time" / 1991 * 2000 /;
```

  - This means:

```
Set t "time" /1991,1992,1993,1994,1995,1996,1997,1998,1999,2000/;
```

  - If the only characters that differ are digits then a label is constructed for every integer in the sequence

```
Set g1 /a1bc*a20bc/;
```

  - But

```
Set crops /wheat * cott/;
```

=> ****Error

GAMS Statement: Alias

- Sometimes it is necessary to have more than one name for the same set

```
SET COLS /wheat, barley, rapeseed, sugarbeet/;

Alias (COLS, CACT);
```

- The newly introduced set name may be used as an alternative name for the original set; the associated set will always contain the same elements as the original set.
- The order of the set names in the alias statement does not matter.

Subsets

- It is often necessary to define sets whose members must all be members of some larger set:

```
SETs   COLS          /wheat, barley, rapeseed, sugarbeet/

       export(COLS)  /wheat, barley/ ;
```

- Where

SETs	Keyword for set definition
export	A subset of the larger set
(COLS)	The original set also called superset
/wheat,barley/	Set elements of the subset

- All elements of the subset must also be elements of the superset.

Introducing Sum in the GAMS model

GAMS Statement: Sum

- Example: to sum all values of X over an index i element of set i={1,2,3,4,5}, in formula notation:

$$Y = \sum_{i=1}^{5} X_i$$

- In GAMS, this can be written as follows:

```
Set I /1,2,3,4,5/ ;
Parameters X(I), Y;
Y = sum (I, X(I));
```

GAMS Statement: Sum

- The general syntax is:

```
sum(settovary,expression)
```

- Where

  `settovary` the name of the set or sets that will be varied

  `expression` generally a function of the set in the sum

- When more than one set is to be varied they are enclosed in parentheses:

```
sum((i,j),x(i,j))
```

Exercise 3. Introduce Set and Sum to Myfarm Example

- Modify the previous model:
  - ✓ Introduce a set “COLS”
  - ✓ Introduce parameters “p_uvag” and “p_lab”
  - ✓ Let these run over the set “COLS”
  - ✓ Let the variable v_actlevl run over the set “COLS”
  - ✓ Modify the equations such that you have a sum statement in each of the equations

Further Useful GAMS Statements

GAMS Statement: Prod

- Products are defined in GAMS using exactly the same format as summations, replacing `Sum` by `Prod`
  - $Y = \prod_i X_i$
- In GAMS, this can be written as follows:

```
Y = prod(I, X(I));
```

GAMS Statement: Table

- Tabular data can be declared and initialized in GAMS using a `table` statement.

```
Sets  COLS       /wheat, barley, rapeseed, sugarbeet/
      ROWS factor inputs    /land, lab/;
Table req(ROWS,COLS) input requirements per ha
                wheat    barley   rapeseed   sugarbeet
    land          1         1          1          1
    lab           25       36         27         87
    ;
```

Set representing the rows

Set representing the columns

- Where

  `Table` Keyword for table definition

Variable Attributes

- While a GAMS parameter has one number associated with each unique label combination, a variable has several
  - ✓ .lo — Lower bound for the variable
  - ✓ .up — Upper bound for the variable
  - ✓ .fx — A fixed value for the variable
  - ✓ .l — Activity level for the variable, also the current value or starting point. This attribute is reset to a new value when a model containing the variable is solved.
  - ✓ .m — The marginal value (or reduced cost) for the variable. This attribute is reset to a new value when a model containing the variable is solved.

Variable Attributes

- For example, in the Myfarm model, fix the area devoted to wheat 60 ha:

```
Positive variable
  v_actLevl(crops)  land area planted with crop
  ;
  v_actLevl.fx("wheat") = 60
  ;
```

GAMS Statement: Loop

- The `loop` statement facilitates executing a group of statements for each member of a set.
- The syntax of the `Loop` statement is

```
Loop((sets_to_vary),
Statement/statements to execute);
```

- Example

```
set Y  'years'   /2011*2020/ ;
parameter pop(y)    'population'  /2011 100/
          grate(y)  'growth rate';
          grate(y) = 0.02;
loop(y, pop(y+1)  = pop(y) *(1+grate(y)));
display pop;
```

Data Exchange with Microsoft Excel

Data Exchange in GAMS

- GAMS can communicate with Microsoft Excel via GDX (GAMS Data Exchange) files

- A GDX file is a file that stores the values of one or more GAMS symbols such as sets, parameters variables and equations.
- A GDX file does not store a model formulation or executable statements.

From GAMS to Excel

1. Writing the data into a GDX file (during the execution time)

```
execute_unload "results.gdx" v_actLevl, v_obje
```

Where

`execute_unload`	GAMS statement to create a GDX file containing selected problem data
`"results.gdx"`	The Name of the GDX file to be created
`v_actlevl, v_obje`	Items written to GDX file

✓ Without specifying any identifier, all sets, parameters, variables and equations will be written to the GDX file.

- Implement the code in the Myfarm model and open the GDX file in GAMSIDE.

From GAMS to Excel

2. Converting the GDX file to Excel file

```
execute 'gdxxrw.exe results.gdx o=results.xlsx var=v_actLevl.L rng=sheet1!A1'
```

- Where

`Execute`	GAMS keyword
`gdxxrw.exe`	Utility to read and write Excel spreadsheet data
`results.gdx`	GDX input file generated by step 1
`o=results.xlsx`	Excel output file
`var=`	Specify the variables you want to export. In case of parameters use `par=`
`rng=sheet1!A1'`	Excel sheet name and for cell range to be used. If skipped, the data is written in cell A1 and beyond in the first available sheet.

Exercise 4. From GAMS to Excel

- Write the variable levels to Sheet1 of the Excel file called „MyFarm_results.xlsx“
- Write the marginal values of the variables to Sheet2 of the same Excel file.

- Question
  - Why are only the values for barley and sugarbeet exported?

From Excel to GAMS

1. Unload the data from Excel to GDX file

```
$call GDXXRW data.xlsx o=data.gdx par=parname rng=sheet1!B3:F5
```

	A	B	C	D	E	F
1	MyFarm Data					
2						
3			wheat	barley	rapeseed	sugarbeet
4		uvag	253	443	284	516
5		lab	25	36	27	87
6						

- Where

`$call`	The command to execute a program called GDXXRW
`GDXXRW`	The utility to read the .gdx file
`data.xlsx`	The excel file located in the same project directory
`o=data.gdx`	The generated .gdx file
`par=parname`	Parameter to be loaded to the GDS file
`rng=sheet1!B3:F5`	Range of the data in Excel

From Excel to GAMS

2. Read in the data from the GDX file

```
Parameter
Data(*,cols)
$GDXIN data.gdx
$LOAD or $LOADdc data
$GDXIN
gm(cols) = data("uvag",cols);
lab(cols) = data("lab",cols);
```

- Where

$GDXIN	command used in a sequence either to load specified items from a GDX file or to close the specified GDX file.
$LOAD	command loads specified items from a GDX file.
$LOADdc	domain check: element names being loaded are in the associated sets. In contrast, the $load command simply ignores elements that are not in the domain

Exercise 5. From Excel to GAMS

- Create an Excel file and call it data.xlsx
- Include the required data into data.xlsx
- Instead of defining the parameter values in GAMS, import them from data.xlsx