Introduction to GAMS Modeling Language

Part I

Dr. Alexander Gocht

Dr. Davit Stepanyan

Thuenen Institute of Farm Economics, Braunschweig

Structure of the Tutorial (part I)

- GAMS modeling language. General introduction; Introduction to GAMS IDE and Studio
- MyFarm LP model; Main elements of a GAMS model: variables, parameters, equations
- Coding MyFarm LP model in GAMS; Solver output
- Solver output; Improving efficiency. Sets, Subsets, Alias, Sum
- Introducing Sets and the Sum operator in MyFarm LP; Further useful Gams statements. Prod, Table, Variable-Attributes, Loop

Objectives

- Upon completion of this tutorial, the participants will be able to:
  - Use the GAMS Studio 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?

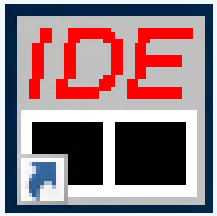
- General Algebraic Modeling System (GAMS) is a high-level modeling system for mathematical programming and optimization.
- GAMS is “a tool for the development, solution, and management of large scale optimization problems”
- Their main distinguishing features are :
  - the use of relational algebra
  - 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

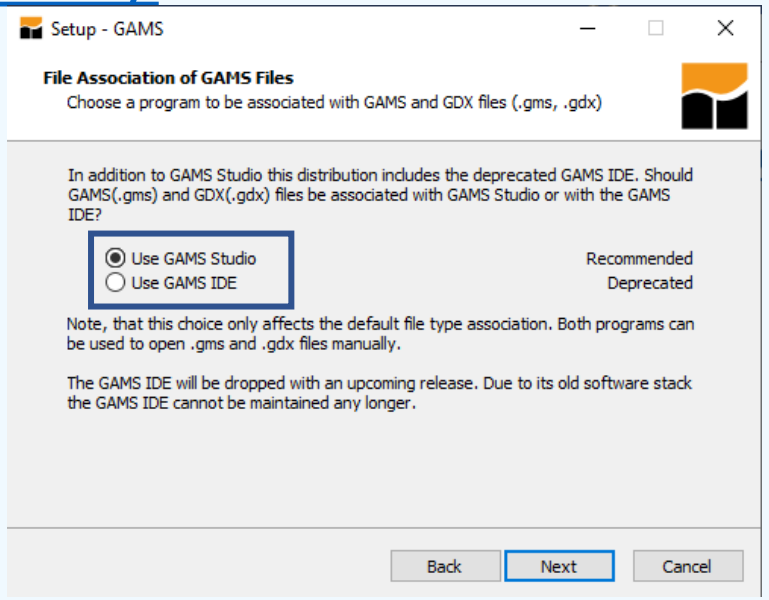
- GAMS IDE
  - Written in Delphi
  - In use for + 20 years
  - Restricted to Windows



- GAMS Studio
  - Written in C++
  - Since 2019
  - Setup similar to IDE
  - Platform-independent (Windows, macOS, Linux)

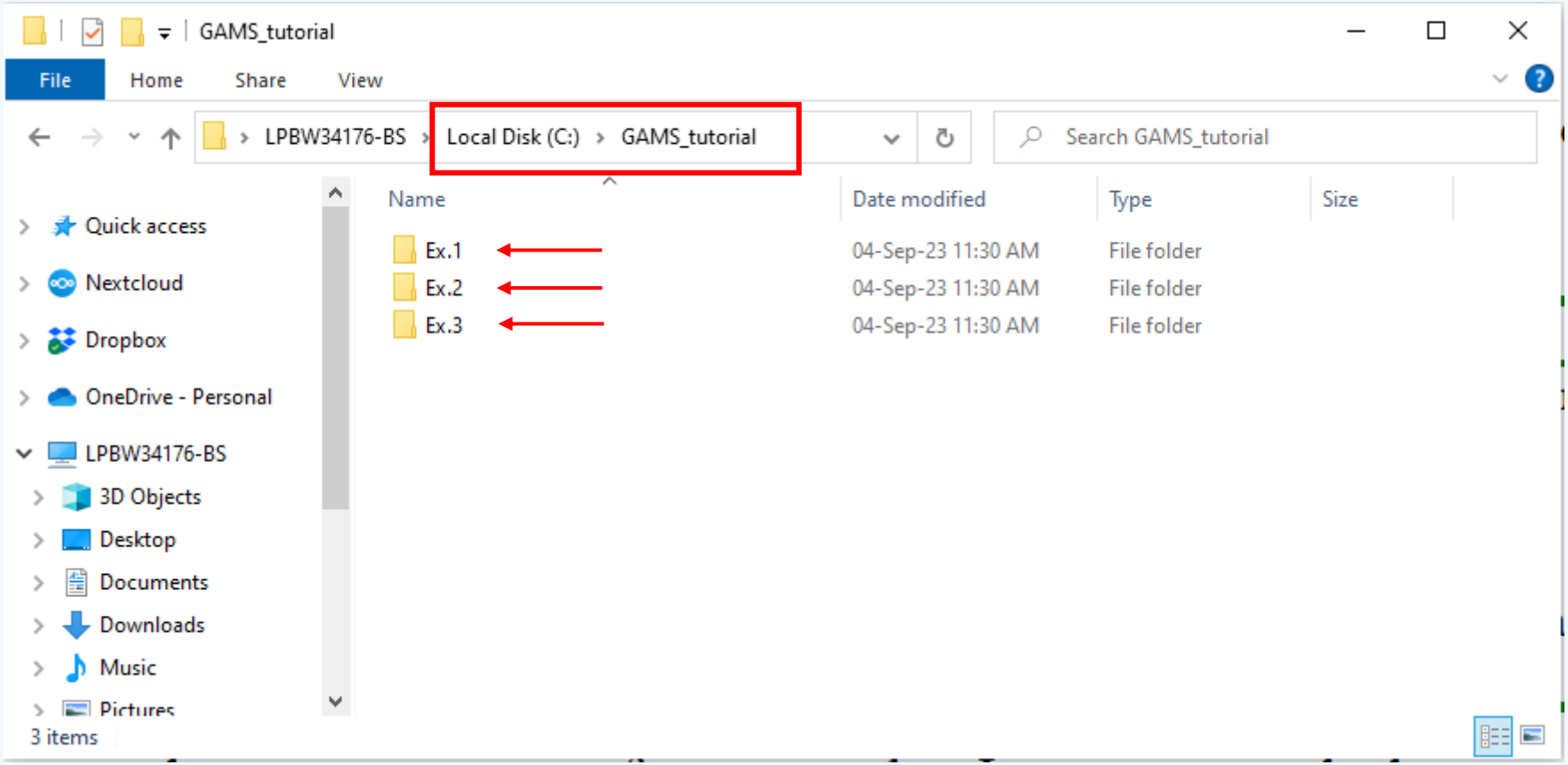
Installing GAMS

- Download the latest GAMS release:
  https://www.gams.com/download/
- Install GAMS
  - Directory: C:\GAMS\
  - Choose GAMS Studio as default
  - Open the GAMS license file distributed with CAPRI and copy containing text on clipboard: "..\GUI\gams_exe\42\gamslice.txt"



- Create a directory for our class projects:
  - C:\GAMS_Tutorial\

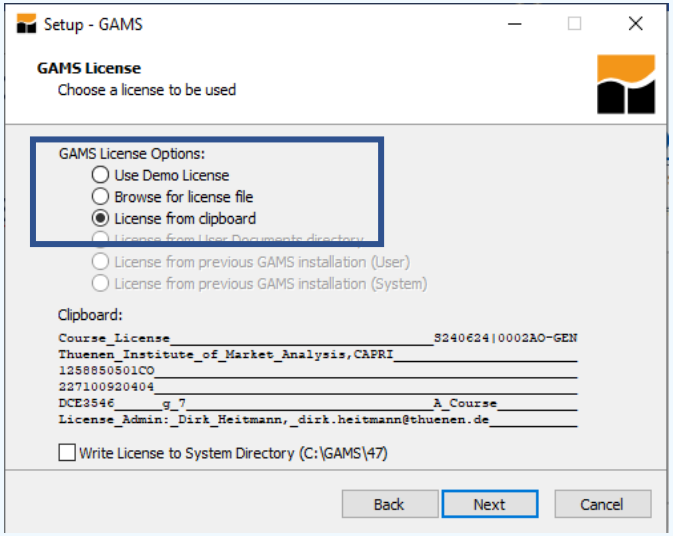
Organization of the Files

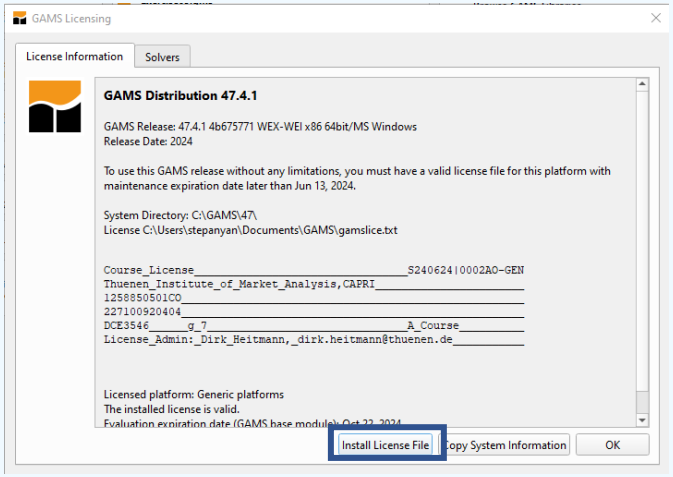


Installing GAMS

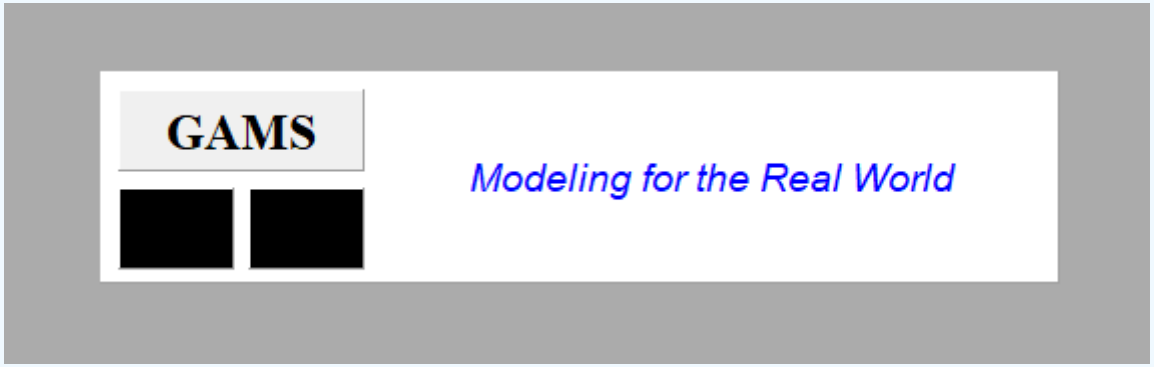
- License file content copied to clipboard is automatically recognized during installation.
- To install the license after GAMS Studio is installed:
  1. Open GAMS Studio
  2. Navigate to Help -> GAMS Licensing -> Install License File
  3. Select the file from the directory:

  “..\GUI\gams_exe\42\gamslice.txt"





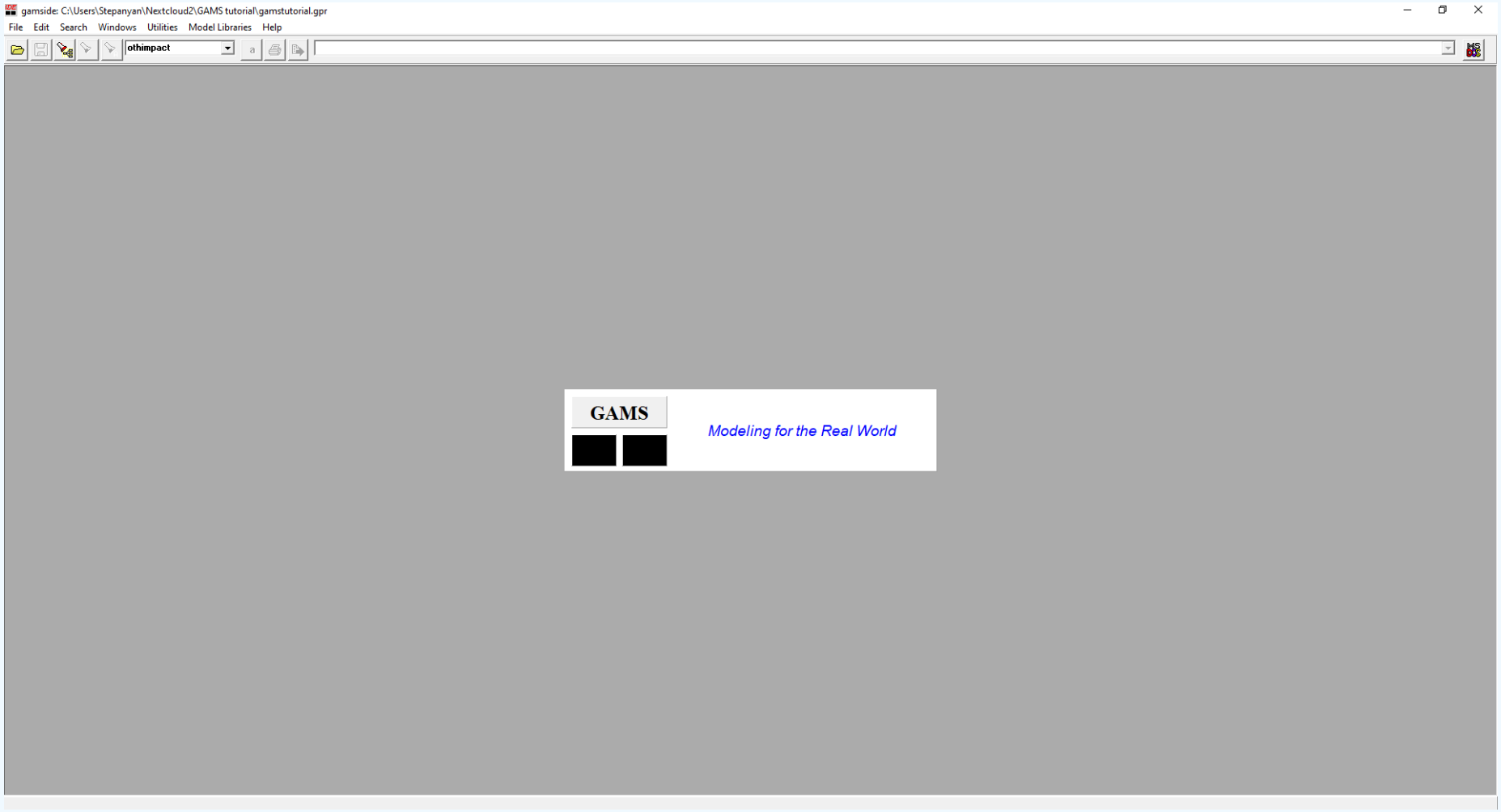
GAMS IDE
(Integrated Development Environment)
Introduction



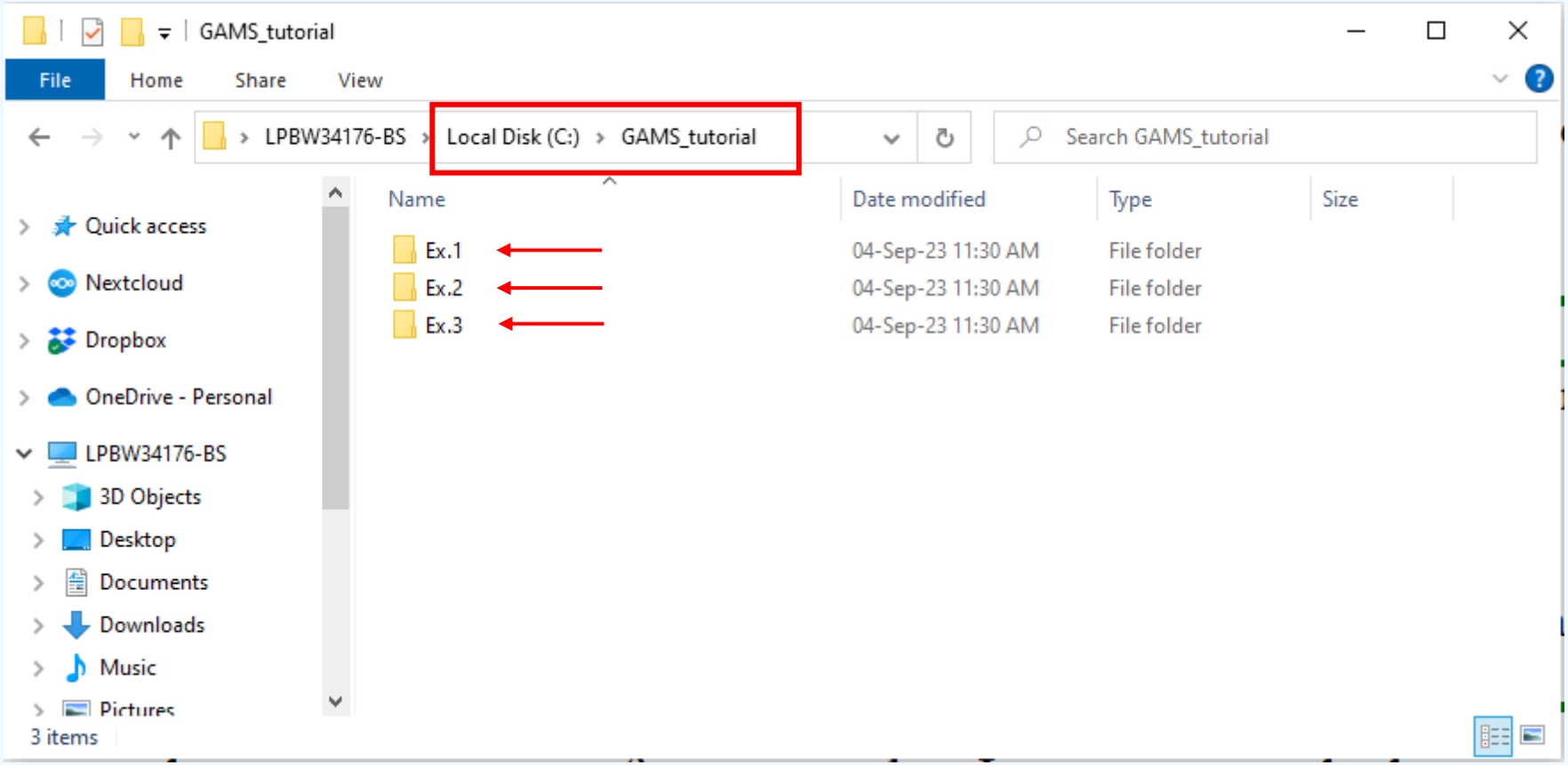
GAMS IDE

- 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

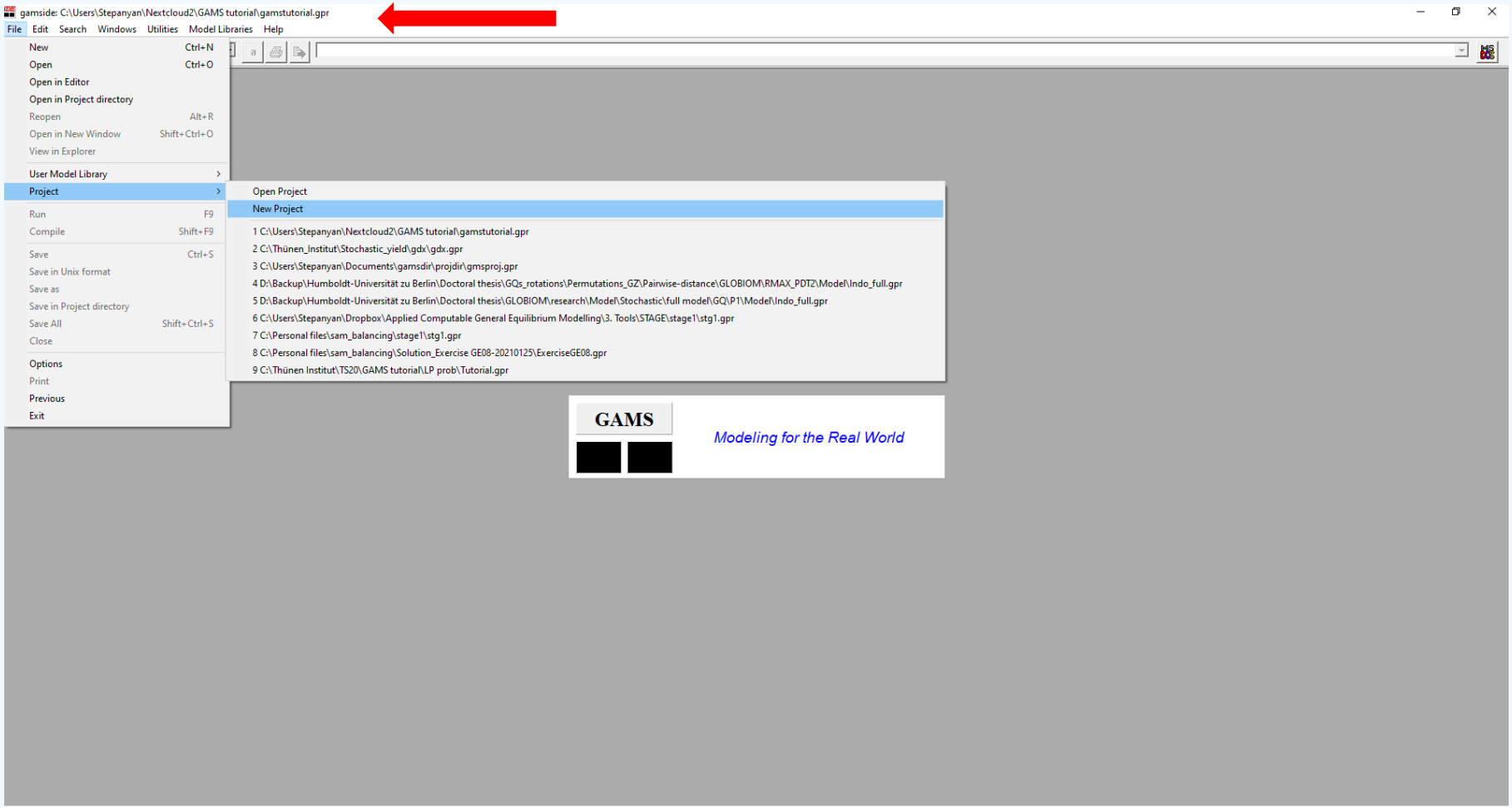
GAMS IDE


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

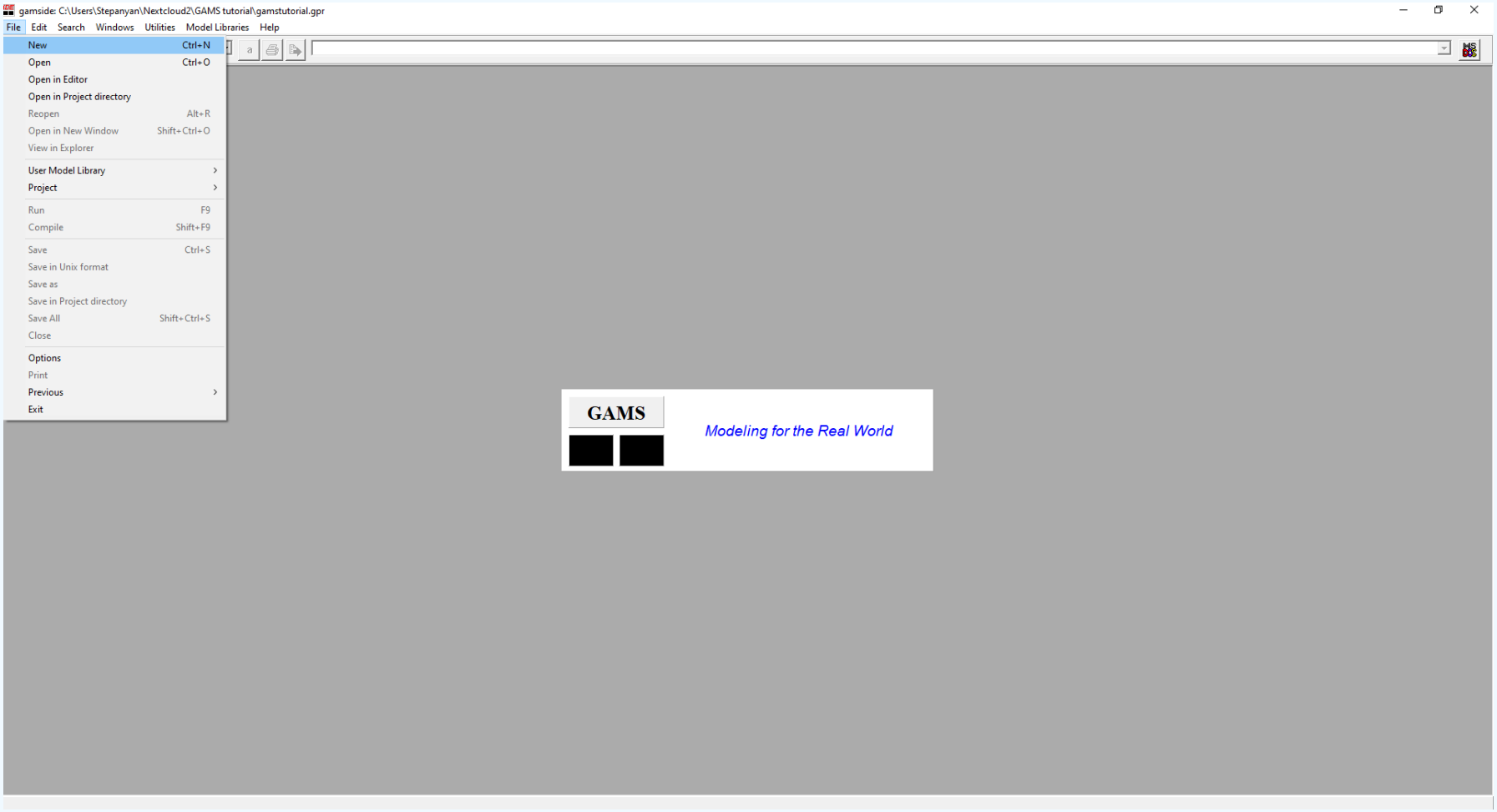
Organization of the Files



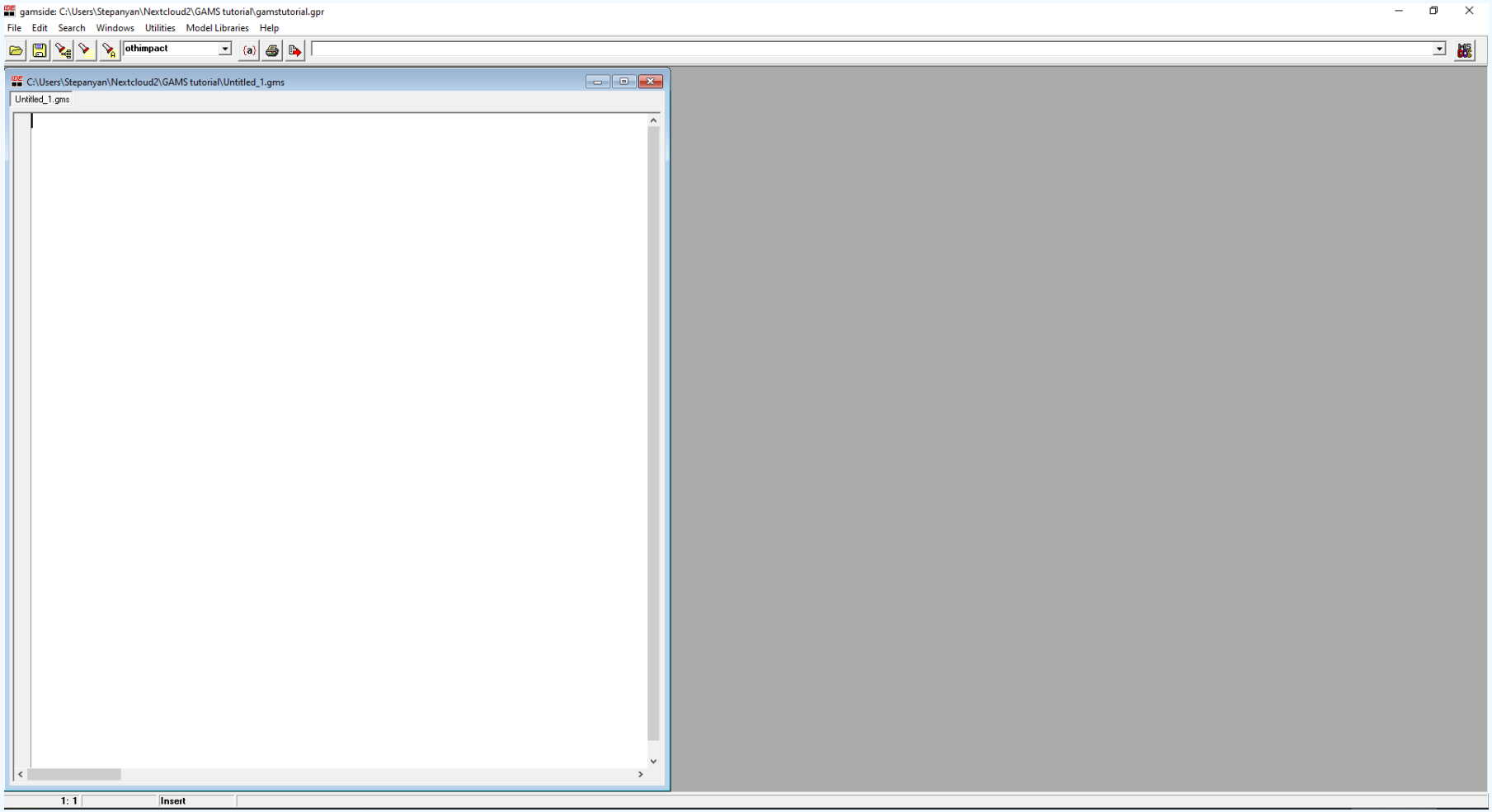
GAMS Project File (.gpr)



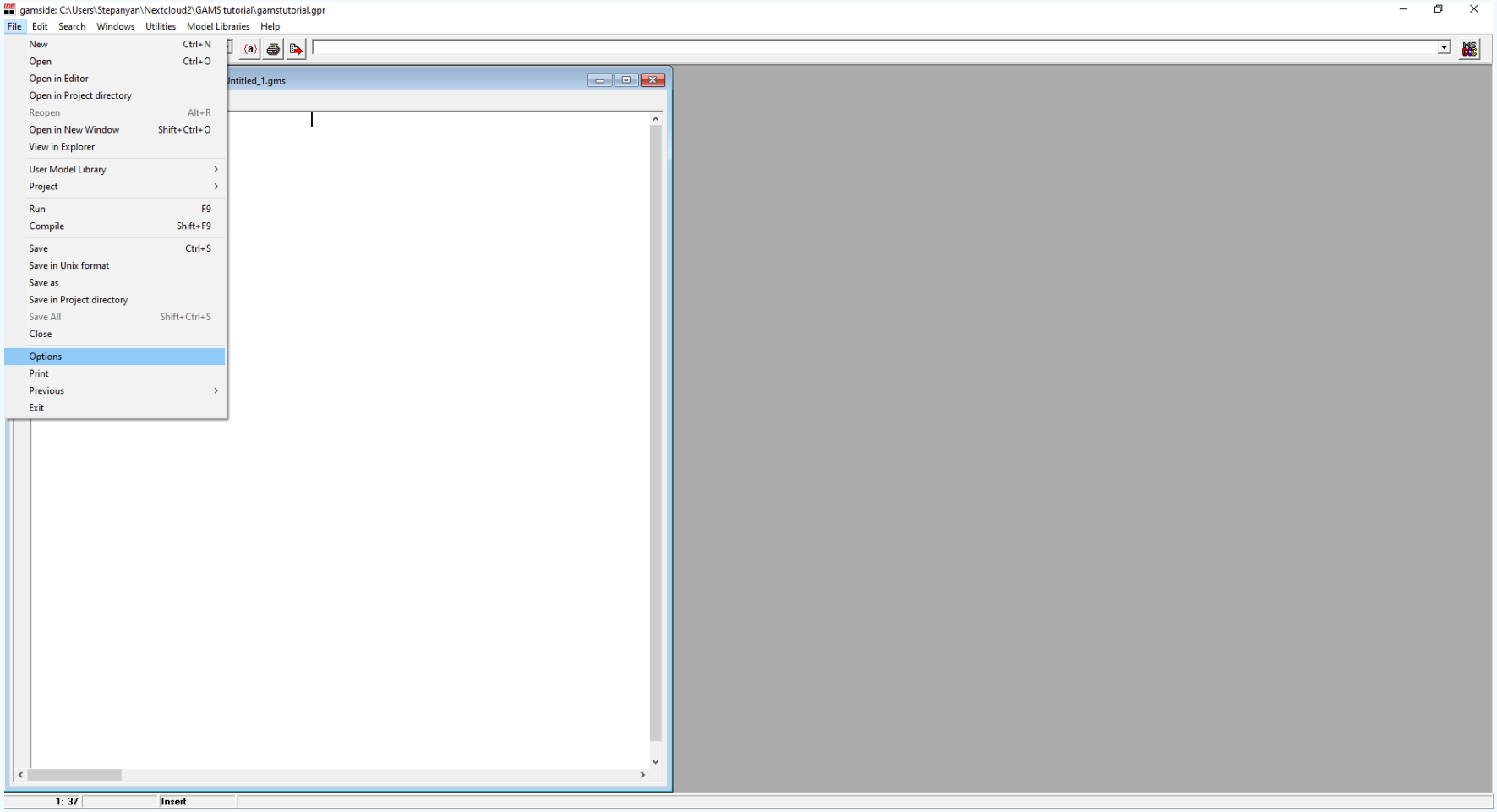
GAMS Files (.gms)



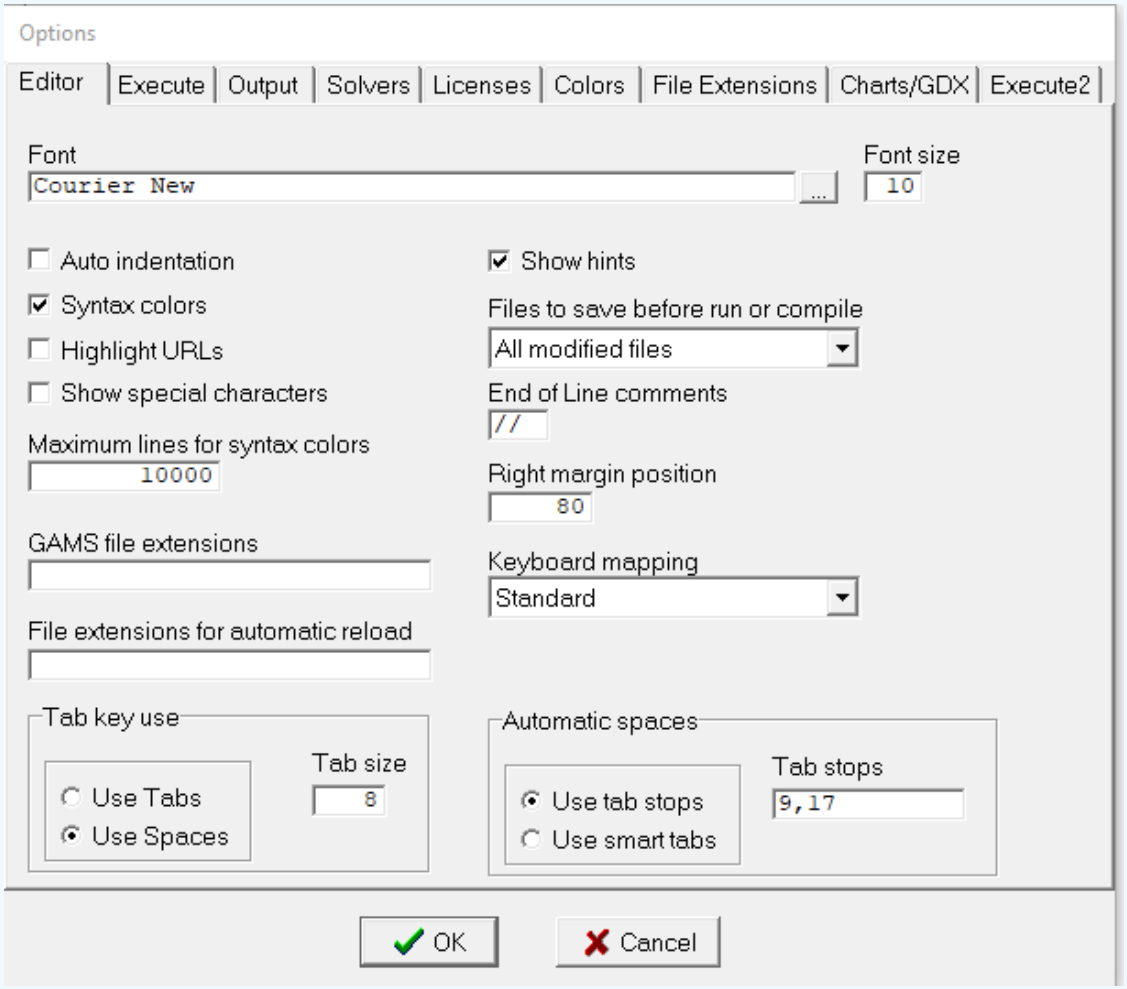
GAMS Files (.gms)



GAMS IDE. Useful Features

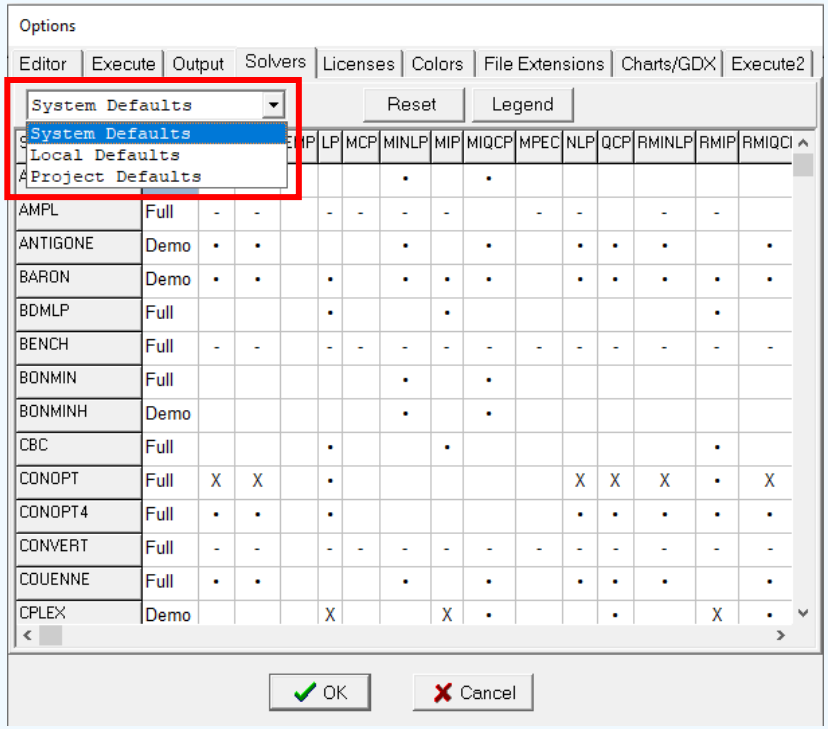


GAMS IDE. Useful Features



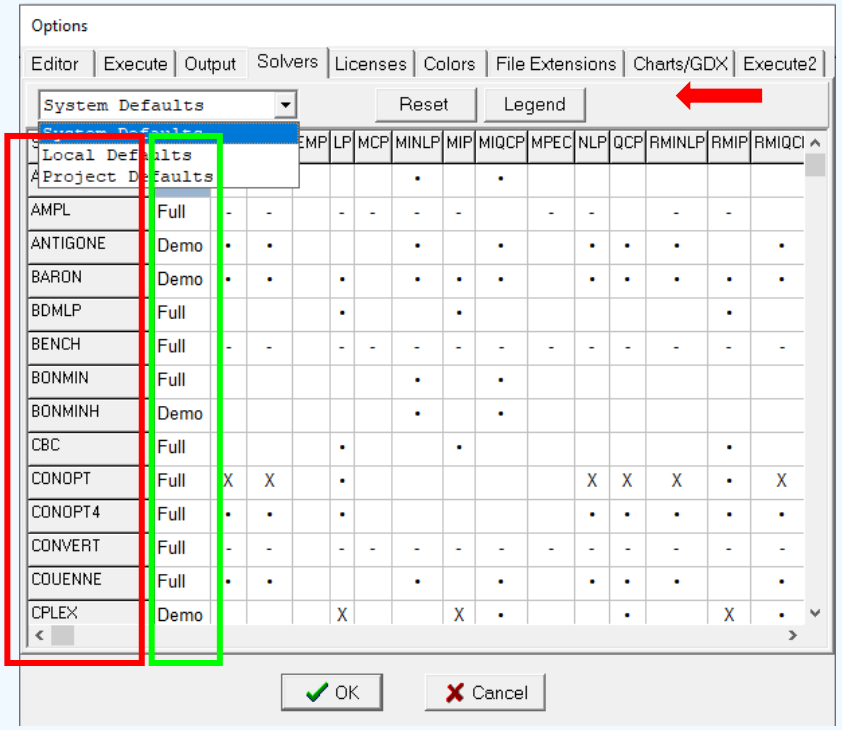
GAMS IDE. Useful Features

- 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

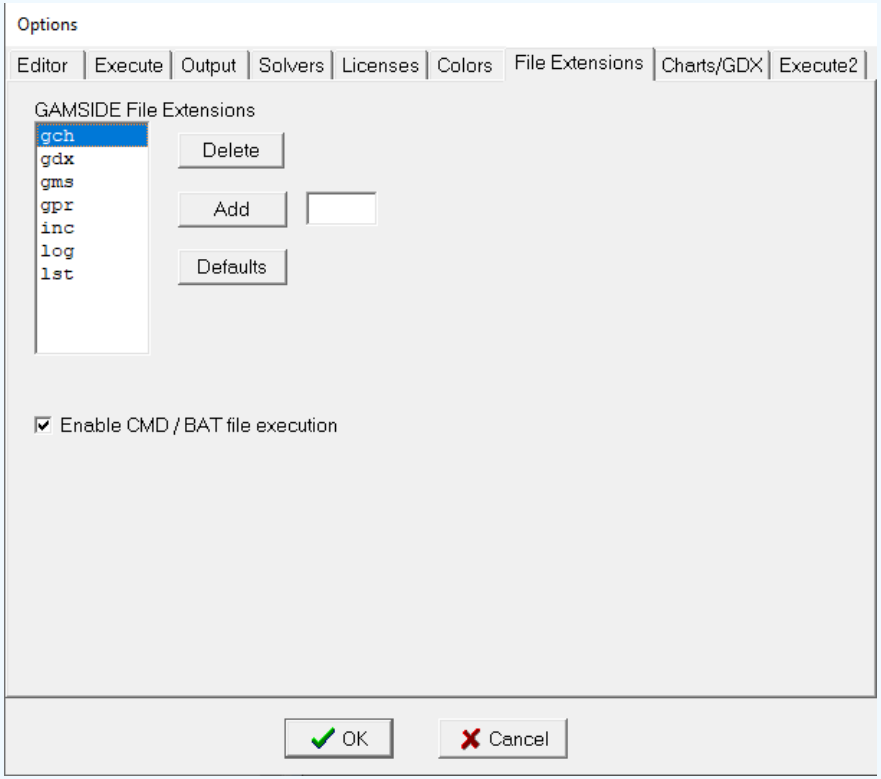
X Current selection

• Available selection

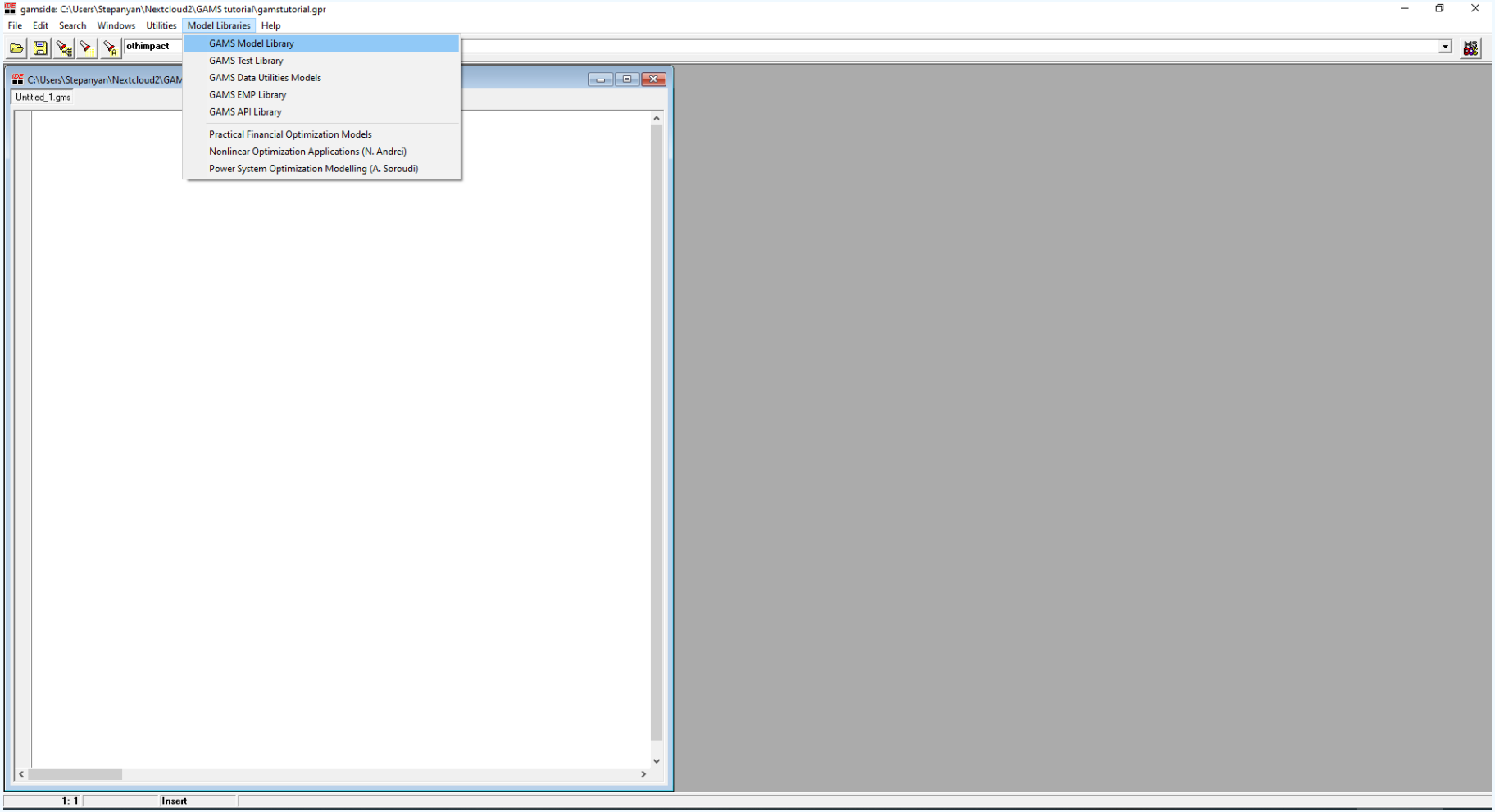
- Not available (Option statement only)

GAMS IDE. Useful Features

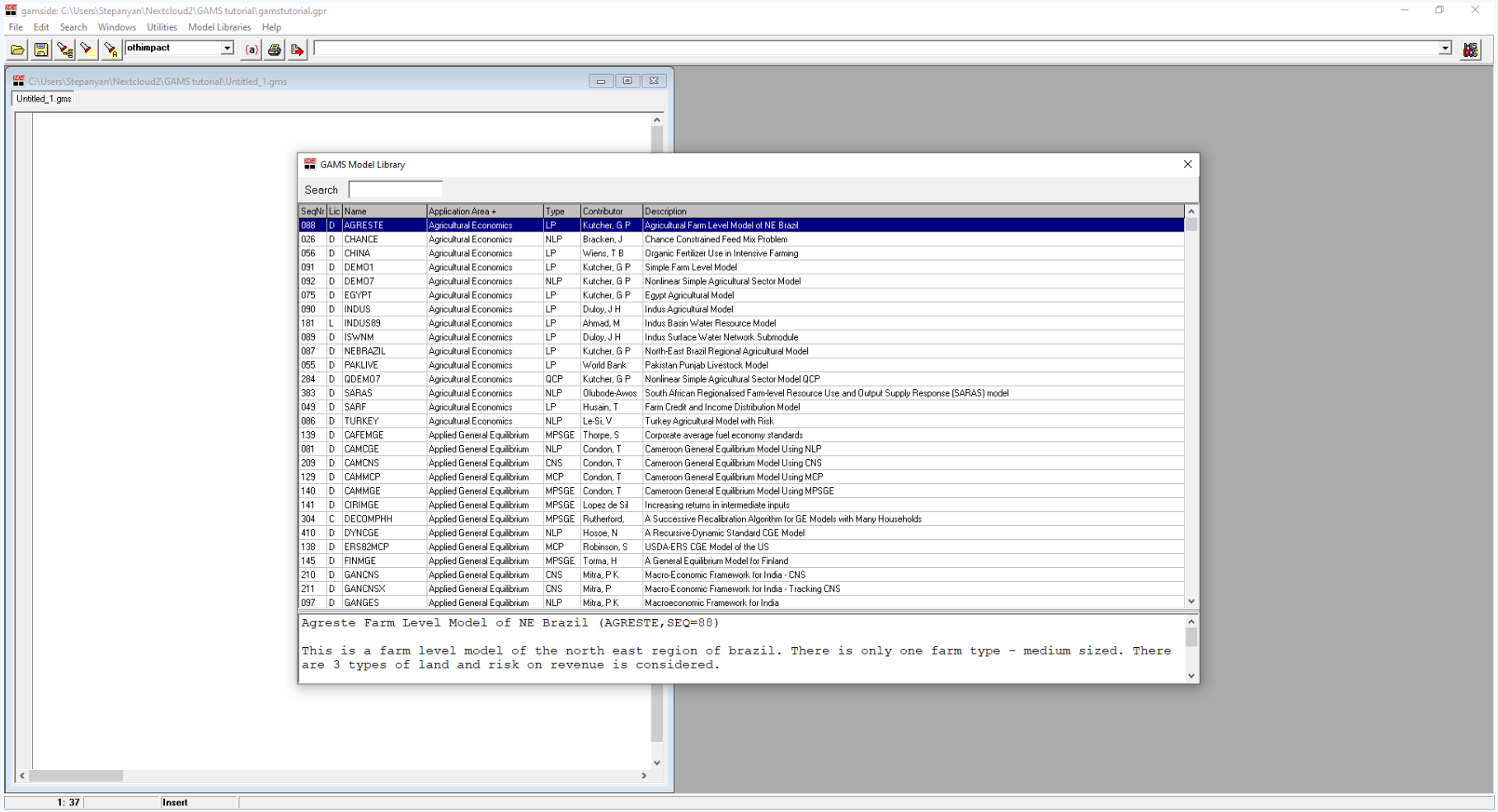
- File Extensions:
  - Select “Defaults”
  - Replace all file extensions with the GAMS standard file extensions ('gms', 'gpr', 'log' and 'lst’).
  - Associate file extensions with the GAMS IDE.



GAMS IDE. Useful Features

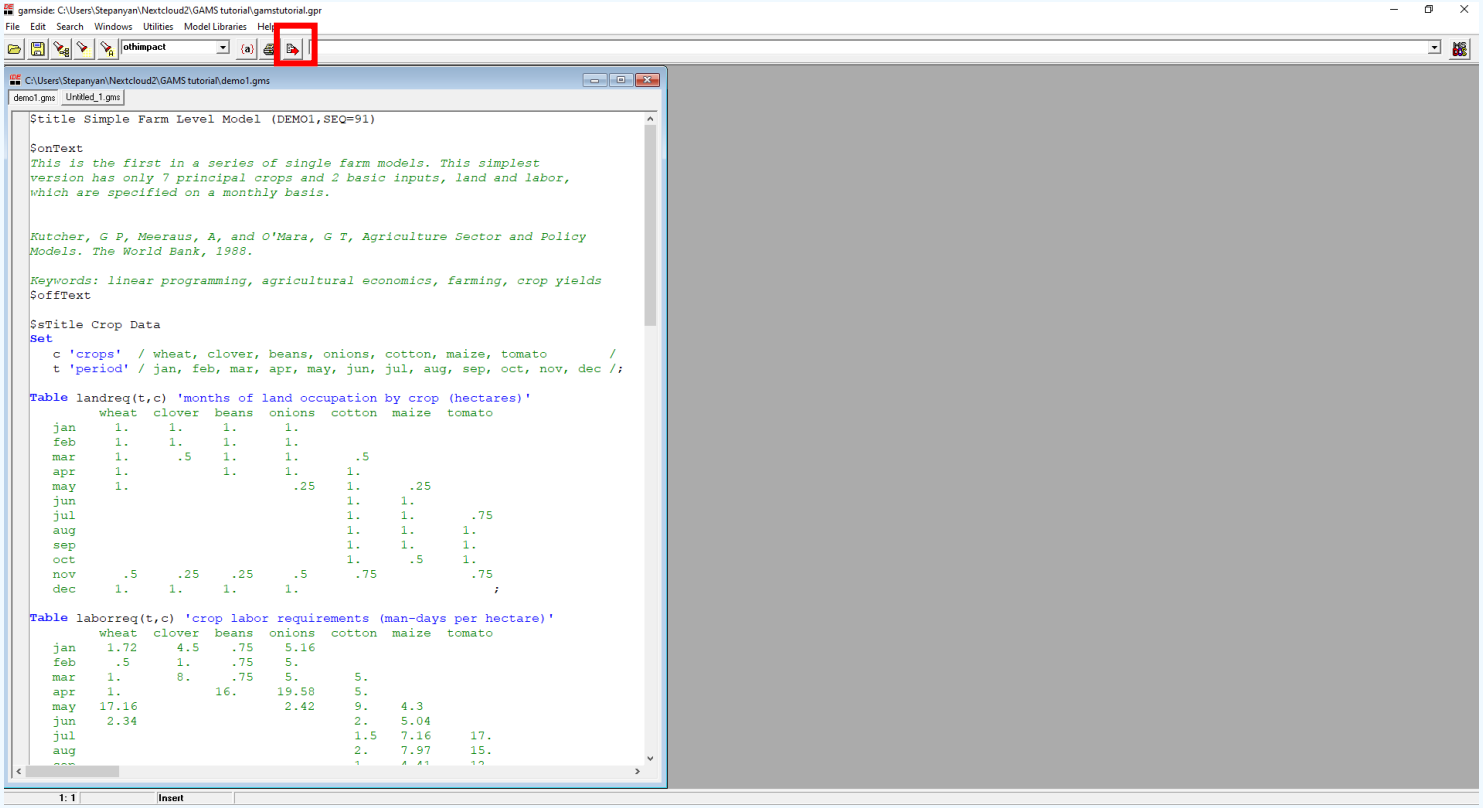


GAMS IDE. Useful Features



Running a Model in GAMS IDE

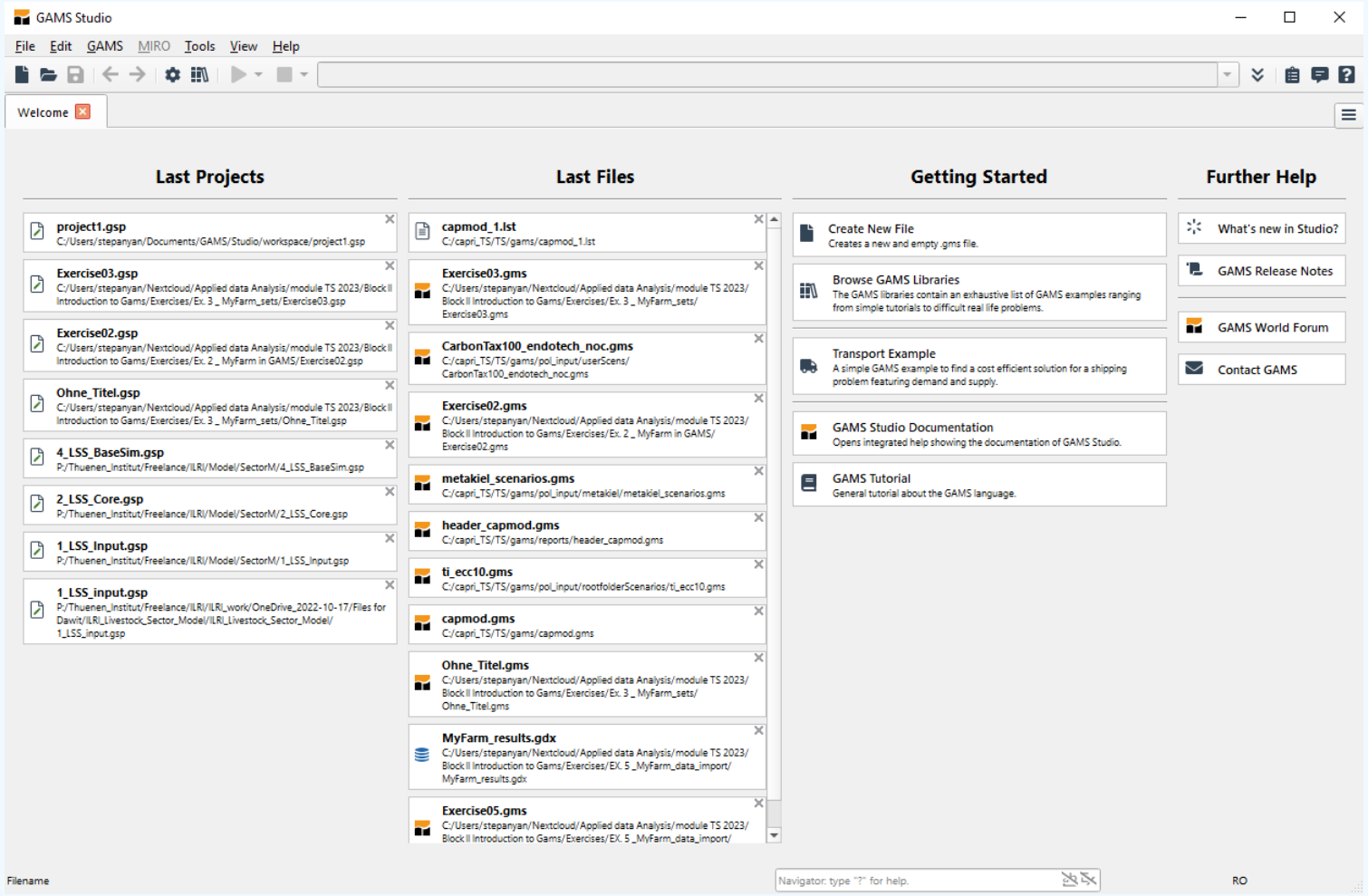
- Or simply press F9



GAMS Studio

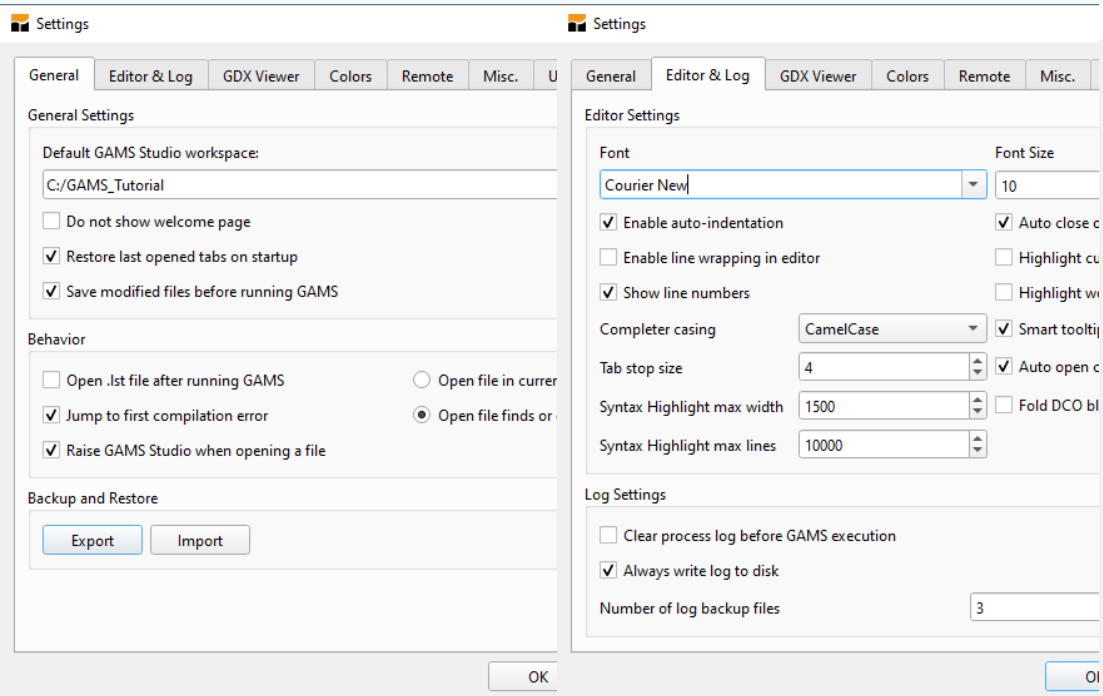
Introduction

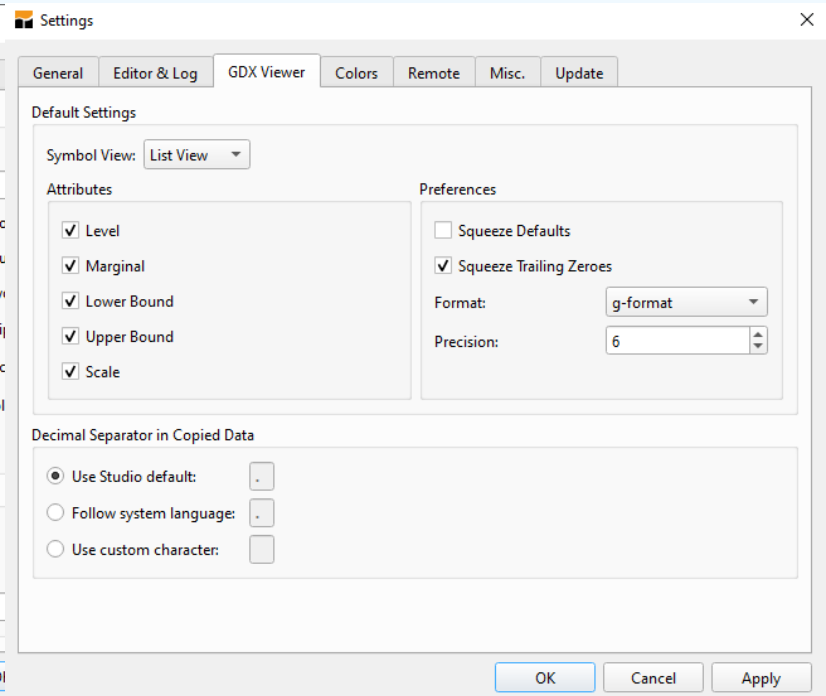
GAMS Studio: Welcome Page



GAMS Studio: General Settings

- File -> Settings or [F7]





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/ Compilation
  - Error messages
  - Equation listing
  - Column (variable) listing
  - 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”

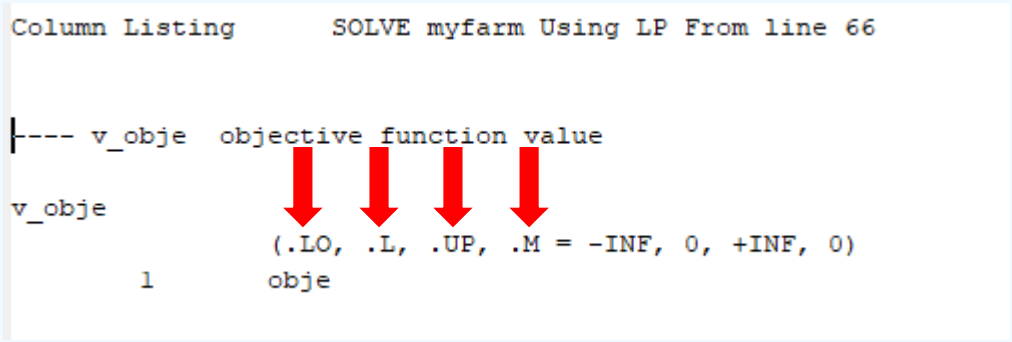
- Is the first part of the output generated by a solve statement
- By default, the first three equations in every block are listed
  - This can be modified with the option `limrow`

```
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;
```

Introduction to GAMS Modeling Language

Part II

M.Sc. Ferike Thom

Thünen Institute of Farm Economics, Braunschweig

Exercise 4. Introducing common income support schemes

Extending the MyFarm Model

Common agricultural income support schemes

Why do farmers receive income support?

The European Union provides farmers with income support to:

- function as a safety net and make farming more profitable;
- guarantee food security in Europe;
- assist them in the production of safe, healthy and affordable food;
- reward farmers for delivering public goods not normally paid for by markets, such as taking care of the countryside and the environment.

Source: European Commision (2022)

Types of income support

- (Voluntary) coupled income support payments (VCS)
- Decoupled income support payments (DIS)
- Quotas

- Ex 4 - Scenario analysis
- Ex 4.1 - Increase gross margin
- Ex 4.2 - VCS
- Ex 4.3 - Decoupled support (DIS)
- Ex 4.4 - Sugarbeet quota
- Ex 4.5 - Sugarbeet quota & VCS

Today‘s objectives

- Add income support schemes to MyFarm model
- Compare effects of different support schemes on the farm

Main take-aways:

- Learn how to do scenario analysis with an existing GAMS model
- Learn how to implement simple policy measures in a simple farm model

Pre-requisite: Scenario analysis

- Goal: When performing scenario analysis, we need an overview of how the different scenarios affect our main model outcomes
- **Question: What are the main model outcomes of the MyFarm model?**

Pre-requisite: Scenario analysis

	Baseline	Coupled support (VCS)	Decoupled support (DIS)	Quota
Crop shares Wheat Barley Rapeseed Sugarbeet				
Total labor requirements				
Total gross margin				

Pre-requisite: Scenario analysis

- GAMS makes it easy to extend models in order to perform scenario analysis
- After each solve statement, we solely need to change the parameters/variables/equations that change within the scenario and may re-run the model

```
Parameter p_results(*,*);
p_results(cols,"base") = v_actLevl.L(cols);
p_results("Labour", "base") = sum (cols,p_lab(cols)*v_actLevl.L(cols));
p_results("Gross Margin", "base") = v_obje.L;
```

Pre-requisite: Scenario analysis

- We may store the results of each model run in a parameter p_results
- Generally, the results parameter does not only contain the results of one particular set (e.g., crops), but rather different ones

  →make use of the „universal set“ * (no domain checking!)

```
* First domain is the result variable (crops, gm, …)
* second domain is the scenario name
Parameter p_results(*,*);
p_results(cols,"base") = v_actLevl.L(cols);
p_results("Labour", "base") = sum (cols,p_lab(cols)*v_actLevl.L(cols));
p_results("Gross Margin", "base") = v_obje.L;
```

Ex 4.1 - Increase gross margin

To Do:

- Introduce a result parameter to your model (after the solve statment)
- Store the model baseline results in your new result parameter
- Change the gross margin of rapeseed to 450€/ha in a new scenario (after the first solve statement)
- Store the results of this scenario („R-Hi“) in the result parameter
- Compare the results!

Ex 4.2 - Voluntary coupled support

- Targeted aid to a specific agricultural sector or sub-sector may be needed if it is facing difficulties
- The VCS scheme aims to prevent the escalation of these difficulties, which could cause abandonment of production and could affect other parts of the supply chain or associated markets
- VCS link (couple) income support payments to certain sectors or products
- **Question: What are the risks?**

Pre-requisite: Scenario analysis

- When we want to introduce or replace an equation, we can create a new model that includes the changes

```
model myfarm / all/;
model myfarm_scenario / myfarm - equation_to_remove
                        + equation_to_add/;
```

Ex 4.2 - Voluntary coupled support

```
* Add VCS payment as a parameter
Parameter p_vcs(cols);
p_vcs("wheat") = 175;
* Define a new objective function including the VCS
Equations
  e_obj_vcs objective function including volunatry coupled support payments;

e_obj_vcs ..   v_obje =E= sum (cols, v_actLevl(cols)*(p_UVAG(cols) + p_vcs(cols)))
;
* Create a new model replacing the objective function with our new one
* and solve the new model
model myfarm_vcs / myfarm - e_obj + e_obj_vcs/;
Solve myfarm_vcs using lp maximizing v_obje;
* store the results
p_results(cols,"VCS") = v_actLevl.L(cols);
p_results("Labour", "VCS") = sum (cols, p_lab(cols)*v_actLevl.L(cols));
p_results("Gross Margin","VCS") = v_obje.L;
```

Ex 4.2 - Voluntary coupled support

To Do:

- Create a new scenario:
- Introduce a new parameter storing the VCS payment for each crop (VCS is 175€/ha, only for wheat!)
- Create a new objective function (equation) which incorporates the VCS
- Create a new model including the new objective function
- Store the scenario results and compare to the baseline!

Ex 4.2 - Voluntary coupled support

Given the current (baseline) state of the MyFarm model

- **Question: How much higher would the gross margin of wheat and rapeseed need to be in order to be economically attractive for the farm?**

Optional To Do:

- Create a loop increasing the subsidy level from 170-175€/ha

Ex 4.3 - Decoupled income support

- In the EU CAP, the link between income support payments and the production of specific products has been progressively removed ('decoupled')
  - Avoids overproduction of certain products
  - Makes sure that farmers are responding to genuine market demand

- **Question: How will the decoupled subsidy affect the economically optimal crop shares compared to the baseline scenario?**

Ex 4.3 - Decoupled income support

```
Parameter p_dis;
p_dis = 105;

Equations
  e_obj_dis objective function including decoupled income support payments
;

e_obj_dis ..   v_obje =E= sum (cols, v_actLevl(cols)*(p_UVAG(cols) + p_dis)) ;

model myfarm_dis / myfarm - e_obj + e_obj_dis /;
Solve myfarm_dis using lp maximizing v_obje;

p_results(cols,"DIS") = v_actLevl.L(cols);
p_results("Labour", "DIS") = sum (cols, p_lab(cols)*v_actLevl.L(cols));
p_results("Gross Margin", "DIS") = v_obje.L;
```

Ex 4.3 - Decoupled income support

To Do:

- Create a new scenario
- Introduce a new parameter storing the decoupled payment (105€/ha)
- Create a new objective function (equation) which incorporates the decoupled income support
- Create a new model including the new objective function
- Store the scenario results and compare to the baseline/VCS!
- Optional: Also store the total amount of subsidies received by the farm in the results parameter. Compare between VCS and decoupled support

Ex 4.4 – Sugarbeet Quota

- For decades, the EU sugar market was highly regulated by production quotas
- Farmers could sell the amount of sugarbeets defined by the quota to a guranteed high price
- System was complex, here we assume that the farm cannot produce beyond the quota

Ex 4.4 – Sugarbeet Quota

To Do:

- Implement a quota in the MyFarm model, given the following assumptions:
  - The farm is allowed to produce 1600t of sugarbeets according to the quota, which we assume is 20ha
  - The guranteed gross margin is 650€/ha
- Consider the land constraint in the main model for a way to implement the quota restriction
- Is the farm better off with the quota (in terms of total gross margin)?

Ex 4.4 – Sugarbeet Quota

```
p_UVAG("sugarbeet")  = 650;
Parameter p_quota(cols);
p_quota("sugarbeet") = 20;

Equation
  e_quota(cols)  quota constraint
;

e_quota(cols)$p_quota(cols) ..  v_actLevl(cols) =L= p_quota(cols);


model myfarm_quot / myfarm + e_quota /;
Solve myfarm_quot using lp maximizing v_obje;

p_results(cols,"QUOT") = v_actLevl.L(cols);
p_results("Labour", "QUOT") = sum (cols, p_lab(cols)*v_actLevl.L(cols));
p_results("Gross Margin", "QUOT") = v_obje.L;
```

Ex 4.5 – Sugarbeet Quota and VCS

```
model myfarm_quot_comb / myfarm_vcs + e_quota /;
p_vcs("wheat") = 250;

Solve myfarm_quot_comb using lp maximizing v_obje;

p_results(cols,"QUOT_VCS") = v_actLevl.L(cols);
p_results("Labour", "QUOT_VCS") = sum (cols,
p_lab(cols)*v_actLevl.L(cols));
p_results("Gross Margin", "QUOT_VCS") = v_obje.L;

display p_results;
execute_unload "all.gdx"
```

Exercise 4: Summary of results

```
display p_results;
execute_unload "all.gdx"
```

	base	R-Hi	VCS	vcs_170	vcs_171	vcs_172	vcs_173	vcs_174	vcs_175	DIS	QUOT	QUOT_VCS
wheat			119						119			180
barley	145			145	145	145	145	145		145	180	
Rape seed		123										
Sugar beet	55	77	81	55	55	55	55	55	81	55	20	20
Labour	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	8.220	6.240
Gross Margin	92.608	95.060	92.697	92.608	92.608	92.608	92.608	92.608	92.697	113.608	92.740	103.540

Introduction to GAMS Modeling Language

Part III

Dr. Alexander Gocht

Thünen Institute of Farm Economics, Braunschweig

slido

Coupled payments of the CAP change

ⓘ Start presenting to display the poll results on this slide.

slido

If we solve a model again

ⓘ Start presenting to display the poll results on this slide.

slido

To read the marginal value of a variable you use

ⓘ Start presenting to display the poll results on this slide.

slido

A model of e_equ1 and e_equ2 can be defined

ⓘ Start presenting to display the poll results on this slide.

Exercise 5: Crop nutrient need

Extending the MyFarm Model

Background/Approach: Nutrient need

- Crops need fertilization. The source of nutrient can come from different sources
- Animal manure, mineral fertilizer, biological fixation
- Many environmental indicators depend on the nutrient balance
- Aim of the exercise is to define:
  - N/P/K nutrient need of our crops
  - Adding three equation which sums the total N/P/K need during model solve
  - Write gams equation more compact

Exercise 5: Crop Nutrient Need

- ✓ Define a new parameter, where the nutrient requirements are stored per crop and fertilizer type
- ✓ please define parameter always with p_ at the beginning, similar e_ and v_ for equation and variables (improves readability)

```
Parameter p_nutrientNeedperCrop(*,cols) "kg per hectar crop need of nutrients";

p_nutrientNeedperCrop("NITF","wheat") =  178;
p_nutrientNeedperCrop("PHOF","wheat") =  77;
p_nutrientNeedperCrop("POTF","wheat") =  151;
p_nutrientNeedperCrop("NITF","barley") =  122;
p_nutrientNeedperCrop("PHOF","barley") =  77;
p_nutrientNeedperCrop("POTF","barley") =  120;
p_nutrientNeedperCrop("NITF","rapeseed") =  218;
p_nutrientNeedperCrop("PHOF","rapeseed") =  119;
p_nutrientNeedperCrop("POTF","rapeseed") =  66;
p_nutrientNeedperCrop("NITF","sugarbeet") =  321;
p_nutrientNeedperCrop("PHOF","sugarbeet") =  178;
p_nutrientNeedperCrop("POTF","sugarbeet") =  280;
```

Exercise 5: Crop Nutrient Need

- ✓ Define three new equations, one for each of the nutrients to multiply the hectare with the nutrient requirements

- ✓ Define a new variable that is responsible to sum up during optimisation the total crop need

```
equation
e_NUTNED_NITF "equation for N",
e_NUTNED_PHOF "for P",
e_NUTNED_POTF "for K";
```

alternative:

```
equation e_NUTNED_NITF "equation for N";
equation e_NUTNED_PHOF "for P";
equation e_NUTNED_POTF "for K";
```

Exercise 5: Crop Nutrient Need

- ✓ The equation should sum over all crops take the optimized hectare and multiply it with the parameter which contains the crop nutrient need.

```
variable v_Nutrientneed(*) "total fertilzer required";

e_NUTNED_NITF..   sum( cols, v_actLevl(cols) *
p_nutrientNeedperCrop("NITF",cols)) =E= v_Nutrientneed("NITF");

e_NUTNED_PHOF..   sum( cols, v_actLevl(cols) *
p_nutrientNeedperCrop("PHOF",cols)) =E= v_Nutrientneed("PHOF");

e_NUTNED_POTF..   sum( cols, v_actLevl(cols) *
p_nutrientNeedperCrop("POTF",cols)) =E= v_Nutrientneed("POTF");
```

Exercise 5: crop nutrient need

- ✓ Define a new model and solve it and report what is the production program and answer the questions

```
model myfarm_Nutrientneed /all/;
solve myfarm_Nutrientneed using lp maximizing v_obje;
```

Q7.1: How much N/P/K in tones is used with the baseline production program

Exercise 5: crop nutrient need

- ✓ We now show the real power of GAMS compared to other programs; We reduce the three similar equations by using additional dimensions for the equations, the dimension is defined over the nutrients using a set FNUT. First we define the nutrients as a set.

```
set FNUT / NITF          "Nitrogen in fertiliser"
           PHOF          "Phospate in fertiliser [P2O5]"
           POTF          "Potassium in fertiliser [K2O]" /;
```

Define an equation but with one dimension for FNUT and a new model

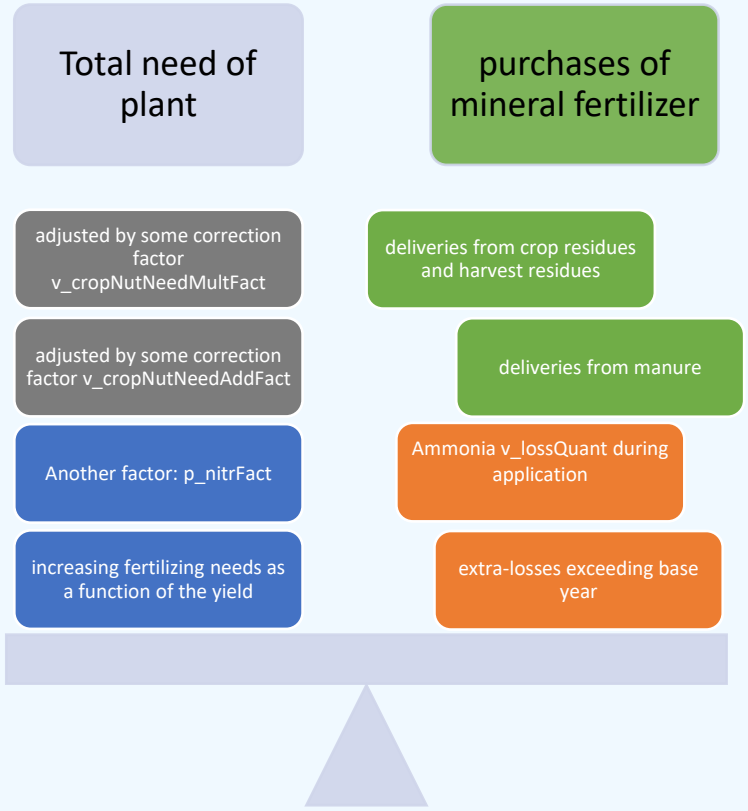
```
equation e_NUTNED_(FNUT);

e_NUTNED_(FNUT)..   sum( cols, v_actLevl(cols) *
p_nutrientNeedperCrop(FNUT,cols)) =E= v_Nutrientneed(FNUT);


model myfarm_Nutrientneed_compact /all - e_NUTNED_NITF -e_NUTNED_PHOF -
e_NUTNED_POTF/;
solve myfarm_Nutrientneed_compact using lp maximizing v_obje;
```

Exercise 5: crop nutrient need

- Q7.2: Did the N/P/K in tones is used with the baseline production program changed?
- Q7.3: Do a new simulation and restrict the nutrient requirements for NITF by 10%. How did the production program change? Hint: variables can be upper bounded during solution like

```
v_Nutrientneed.up("NITF") =  v_Nutrientneed.l("NITF")*0.9;
solve myfarm_Nutrientneed using lp maximizing v_obje;
p_results(cols,"compact_10") = v_actLevl.l(cols);
p_results(FNUT,"compact_10") = v_Nutrientneed.l(FNUT);
*reset the upper bound to positive infitity
v_Nutrientneed.up("NITF") = +inf;
```

Nutrient balance in CAPRI



```
NUTNED_(RUNR,NGRP,FNUT) $ SUM( (CACT_TO_NGRP(MCACT,NGRP),A) $ [p_technFact(RUNR,MCACT,"LEVL",A) and %data%(RUNR,MCACT,"levl","Y")], 1) ..

SUM( (CACT_TO_NGRP(MCACT,NGRP),A) $ p_technFact(RUNR,MCACT,"LEVL",A), v_actLevl(RUNR,MCACT,A)
*
* --- total nutrient need of crops (retention - biolog. fixation for certain crops) * nutrient factor
*
* (%data%(RUNR,MCACT,FNUT,"Y")
* (1-p_nitrFact(RUNR,MCACT,"BioFix") $ SAMEAS(FNUT,"NITF")
)
* v_cropNutNeedMultFact(RUNR,FNUT,A)
*
* --- plus constant term of nutrient factor
*
+ v_cropNutNeedAddFact(RUNR,FNUT))
*
* --- soil property effect
*
* (p_nitrFact(RUNR,"ALL","DEFR") $ SAMEAS(FNUT,"NITF") + 1 $ (NOT SAMEAS(FNUT,"NITF")))
*
* --- increasing fertilizing needs as a function of the yield
*
* SQRT(
[1.+p_technFact(RUNR,MCACT,"Yield",A)
+ 0.2 $ SAMEAS(MCACT,"GRAI") - 0.2 $ SAMEAS(MCACT,"GRAE")
] $ %data%(RUNR,MCACT,FNUT,"Y")
)
) =E=
*
* --- purchases of anorganic fertiliser minus Ammonia v_lossQuant during application
*
+ v_fertDist(RUNR,NGRP,FNUT,"Mine") * (1.-p_emiLoss(RUNR,"NETF","N","GasRunTot") $ SAMEAS(FNUT,"NITF") )
*
* --- deliveries from manure
+ v_fertDist(RUNR,NGRP,FNUT,"Excr") * SUM( FOUT_T_N(FOUT,FNUT),v_nutAvailFactExcr(RUNR,FOUT,"T"))
*
* --- deliveries from crop residues (distribution across arable crops)
*
+ (v_fertDist(RUNR,NGRP,FNUT,"Cres")
* SUM( FOUT_T_N(FOUT,FNUT),v_nutAvailFactCRes(RUNR,FOUT,"T"))
* (1.-p_emiLoss(RUNR,"NETF","N","GasRunTot") $ SAMEAS(FNUT,"NITF") )) $ (NOT PERM_NGRP(NGRP))
*
* --- delivery from harvest residues and atmospheric deposition (only for grass land, where it remains)
*
+ SUM( (CACT_TO_NGRP(MCACT,NGRP),FOUT_T_N(FOUT,FNUT),A)
$ (p_technFact(RUNR,MCACT,"LEVL",A) and PERM_NGRP(NGRP)),
v_actLevl(RUNR,MCACT,A)
* %data%(RUNR,MCACT,FOUT,"Y")
* (p_technFact(RUNR,MCACT,FOUT,A)+1.)
* (1.-p_emiLoss(RUNR,"NETF","N","GasRunTot") $ SAMEAS(FNUT,"NITF") )
* v_nutAvailFactCRes(RUNR,FOUT,"T"));
```

Exercise 5: Results

	Nutrientneed	compact	compact_10
barley	145.098	145.098	162.85
sugarbeet	54.902	54.902	37.1505
NITF	35325.5	35325.5	31792.9
PHOF	20945.1	20945.1	19152.2
POTF	32784.3	32784.3	29944.1

Introduction to GAMS Modeling Language

Part IV

Dr. Alexander Gocht

Thünen Institute of Farm Economics, Braunschweig

Exercise 6: Crop nutrient need and fertilizer purchase

Extending the MyFarm Model

Exercise 6: crop nutrient need & fertilizer purchase in MyFarm

- ✓Crop nutrient need is assumed to be covered by fertilizer purchased
- ✓This are costs which can impact the decision of a farmer as certain crops need more and others less fertilizer particular regarding nitrogen fertilizer
- ✓Aim is to introduce prices for N/P/K and include this as additional costs in the objective function of the farmer

Exercise 6: crop nutrient need & fertilizer purchase

- ✓ Let's assume that the requirements need to fulfilled by purchasing fertilizer to a given price.
- ✓ Please define a price parameter for our nutrients

```
parameter p_fertilzer_price(FNUT);
p_fertilzer_price("NITF") = 0.18;
p_fertilzer_price("PHOF") = 0.25;
p_fertilzer_price("POTF") = 0.11;
```

Minus the costs for nutrients

Then define a new objective function

```
equation e_obj_fertlizer "new objective for including costs of fertilizer";
e_obj_fertlizer ..  v_obje =E= sum (cols, v_actLevl(cols)* p_UVAG(cols)) -
sum(FNUT, v_Nutrientneed(FNUT) * p_fertilzer_price(FNUT));
model myfarm_fert /all - e_obj - e_NUTNED_NITF -e_NUTNED_PHOF -
e_NUTNED_POTF/;
solve myfarm_fert using lp maximizing v_obje;
```

Add some code line for reporting crops levels and nutrient need as we have learnt to see the effect in a parameter as scenarios

Exercise 6: Questions to be answered

- Q8.1: How does the cropping program changes when accounting for fertilizer costs? What could be the reason?

Exercise 6: Results

	Nutrientneed	compact	compact_10	fertcost
barley	145.098	145.098	162.85	200
sugarbeet	54.902	54.902	37.1505	
NITF	35325.5	35325.5	31792.9	24400
PHOF	20945.1	20945.1	19152.2	15400
POTF	32784.3	32784.3	29944.1	24000

Exercise 7: fertilizer price shock in MyFarm

- ✓We use a loop statement to simulate different price for fertilizer N and observe the cropping pattern, shadow value changes and total gross margin changes using a parameter and execute the results to GDX

Exercise 7: fertilizer price shock

✓ Please define a set and a parameter used to change the price

```
set steps/1*5/;
parameter p_steps(steps);
```

Please define for each run in the loop a change factor

```
p_steps("1")= 100;
p_steps("2")= 1.;
p_steps("3")= 0.5;
p_steps("4")= 0.1;
p_steps("5")= 0.0;
```

Please define a reporting parameter with three dimensions

```
parameter p_reporting(steps,*,*);parameter p_fertilzer_price_ori(FNUT);
store value
p_fertilzer_price_ori(FNUT)=p_fertilzer_price(FNUT);
loop(steps,
   p_fertilzer_price("NITF") =  p_fertilzer_price_ori("NITF") * p_steps(steps);
   Solve myfarm_fert using lp maximizing v_obje;
   p_reporting(steps,cols,"prod") = v_actLevl.l(cols);
   p_reporting(steps,"total","GM") = v_obje.l;
   p_reporting(steps,cols,"Shadow") = v_actLevl.m(cols) ;
   p_reporting(steps,"price NITF","shock") = p_fertilzer_price_ori("NITF") *
p_steps(steps);
);
execute_unload "all.gdx";
```

Exercise 7: Questions to be answered cnt.

- Q7.1: Simulate a fertilizer tax for nitrogen "NITF" by decreasing and increasing the fertilizer price stepwise
- Q7.2: Draw or show the relationship between total gross margin and the fertilizer tax
- Q7.3: When the fertilizer price for nitrogen increases drastically what happens with the production program

Exercise 7: Results

	prod		GM	Shadow				shock
	barley	sugarbeet	total	wheat	barley	rapeseed	sugarbeet	price N
1				-2986.86	-1785.45	-3677.01	-5337.3	18
2	200		77718	-203.49		-180.84	-5.67	0.18
3	145.098	54.902	80586	-195.81		-170.04		0.09
4	145.098	54.902	83129.4	-188.688		-160.6		0.018
5	145.098	54.902	83765.3	-186.907		-158.239		

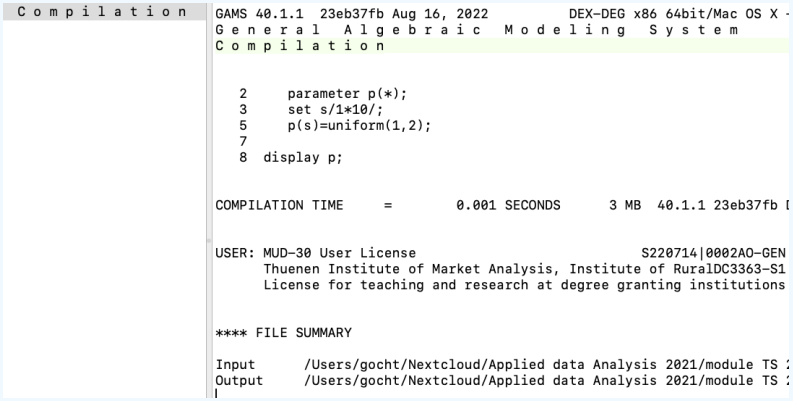
Exercise 8: Compile time conditions

- ✓ When GAMS executes the program the code is first compiled and then executed
- ✓ You can only compile the program by adding “a=c” to the GAMSIDE
- ✓ During compilation it checks (and modify) the code but does not produce any results (does not run any logic) but produces a listing file
- ✓ All command with a $ in the first column of a code-line are compile time statements and are interpreted during the compilation.
- ✓ Compile time statements can not be seen in the .lst file only the the code modifications
- ✓ Dollar control options are not part of the GAMS language they instruct the compiler to perform some task. Therefore, dollar control options are not terminated with a semicolon as real GAMS language statements.
- ✓ See also https://gams.com/latest/docs/UG_DollarControlOptions.html?print=1#UG_DollarControl_Syntax

Exercise 8: Compile time statements

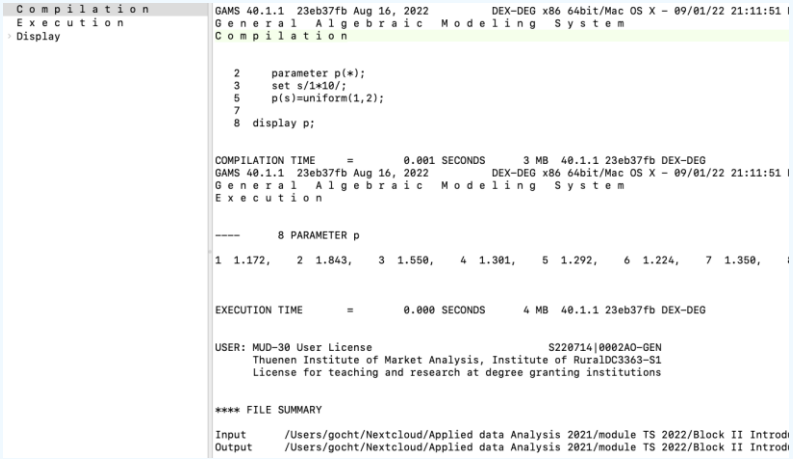
✓ Only Compile (a=c)

```
$setglobal TRIGGER on
                parameter p(*);
                set s/1*10/;
$iftheni %TRIGGER% == on
                p(s)=uniform(1,2);
$endif
                display p;
```



✓ Compile & Execute

```
$setglobal TRIGGER on
                parameter p(*);
                set s/1*10/;
$iftheni %TRIGGER% == on
                p(s)=uniform(1,2);
$endif
                display p;
```



Exercise 8: Compile time statements

✓ Only Compile (a=c)

```
$setglobal TRIGGER off
                parameter p(*);
                set s/1*10/;
$iftheni %TRIGGER% == on
                p(s)=uniform(1,2);
$endif
                display p;
```

```
Compilation

GAMS 40.1.1  23eb37fb Aug 16, 2022          DEX-DEG x86 64bit/Mac OS X - 09/01/22
General Algebraic Modeling System
Compilation


   2     parameter p(*);
   3     set s/1*10/;
   5
   6  display p;
****          $141
**** LINE      8 INPUT       /Users/gocht/Nextcloud/Applied data Analysis 2021/mod
**** 141  Symbol declared but no values have been assigned. Check for missing
****          data definition, assignment, data loading or implicit assignment
****          via a solve statement.
****          A wild shot: You may have spurious commas in the explanatory
****          text of a declaration. Check symbol reference list.

**** 1 ERROR(S)   0 WARNING(S)


COMPILATION TIME     =        0.002 SECONDS      3 MB  40.1.1 23eb37fb DEX-DEG


USER: MUD-30 User License                            S220714|0002AO-GEN
      Thuenen Institute of Market Analysis, Institute of RuralDC3363-S1
      License for teaching and research at degree granting institutions


**** FILE SUMMARY

Input      /Users/gocht/Nextcloud/Applied data Analysis 2021/module TS 2022/Block
Output     /Users/gocht/Nextcloud/Applied data Analysis 2021/module TS 2022/Block

**** USER ERROR(S) ENCOUNTERED
```

✓ Compile & Execute

```
$setglobal TRIGGER off
                parameter p(*);
                set s/1*10/;
$iftheni %TRIGGER% == on
                p(s)=uniform(1,2);
$endif
                display p;
```

```
Compilation

GAMS 40.1.1  23eb37fb Aug 16, 2022          DEX-DEG x86 64bit/Mac OS X - 09/01/22 21:
General Algebraic Modeling System
Compilation


   2     parameter p(*);
   3     set s/1*10/;
   5
   6  display p;
****          $141
**** LINE      8 INPUT       /Users/gocht/Nextcloud/Applied data Analysis 2021/module
**** 141  Symbol declared but no values have been assigned. Check for missing
****          data definition, assignment, data loading or implicit assignment
****          via a solve statement.
****          A wild shot: You may have spurious commas in the explanatory
****          text of a declaration. Check symbol reference list.

**** 1 ERROR(S)   0 WARNING(S)


COMPILATION TIME     =        0.001 SECONDS      3 MB  40.1.1 23eb37fb DEX-DEG


USER: MUD-30 User License                            S220714|0002AO-GEN
      Thuenen Institute of Market Analysis, Institute of RuralDC3363-S1
      License for teaching and research at degree granting institutions


**** FILE SUMMARY

Input      /Users/gocht/Nextcloud/Applied data Analysis 2021/module TS 2022/Block II
Output     /Users/gocht/Nextcloud/Applied data Analysis 2021/module TS 2022/Block II

**** USER ERROR(S) ENCOUNTERED
```

Exercise 8: CAPRI uses a lot of compile time conditions

✓ To copy the content of another file into the program

```
$include 'envind\def_gascoeff.gms'
```

✓ To define variable a simple logic which can be assessed later during compilation

```
$setglobal declAmmoSymbols NO
```

✓ To check of a file exist in a folder and to do simple character manipulation

```
$ifi exist '%results_in%\simini\sim_ini_%NTSLVL%%BAS%%SIM%%MODID%.gdx'
```

✓ They can be combined in one line

```
$ifi %BASELINE%==on $setglobal declAmmoSymbols NO
```

✓ To enforce to ignore parts of the code to be compiled

```
$goto after_simini
 code
$label after_simini
$ontext
 code
$offtext
```

References

- European Commission (2022): Income support explained. https://agriculture.ec.europa.eu/common-agricultural-policy/income-support/income-support-explained_en. Last accessed 26.08.2022.