GAMS-CAPRI Training

Sevilla, 9-11 April 2018

GAMS features

Maria Blanco

Dep. Agricultural Economics

Technical University of Madrid

maria.blanco@upm.es

Overview

- Sets and mappings
- Conditionals
- Initial values and bounds
- Display options
- Comments
- GDX utilities
- GAMS functions
- Control variables
- Conditional compilation
- Model attributes and options
- Partial run

Sets and mappings

Working with SETS – subsets

Subsets contain part of the elements of another set

- All elements of the subset must be elements of the larger set.
- The elements of the subset may be defined explicitly or may be calculated.

Example

```
set rops all crops /wheat,maize,tomato,potato
                    sunflower,soya,sugarbeet/

set cereals(crops) crops /wheat,maize/
;
```

Working with SETS – Alias

Alias statement

- Gives another name to a set defined previously
- Useful in market equilibrium problems to specify cross elasticities

Syntax

```
alias(knownSet,newSet);

* example
set c commodities ;
alias(c,cc);
```

Working with SETS – dynamic sets

Subset (of a static set) whose elements can change

- The keywords used to denote membership or non-membership are YES and NO

Example

```
set
Y  'years'                    /2010*2020/
cury(y) 'current year'
;
  cury('2010') = yes;
```

- Dynamic sets cannot be used as domains

Working with SETS – ord and card

Ord and card

- Ord: parameter that indicates the relative position of each element in the set
- Card: scalar that indicates the number of elements in the set

Example

```
set Y  'years'   /2011*2020/ ;

parameter p_ord, p_card ;

p_ord(y) = ord(y);
p_card   = card(y);

display p_ord, p_card;
```

Working with SETS – lead and lag

Lead and lag

- Lead-lag effect
- Links between variables over time

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;
```

Working with SETS – mappings

Multidimensional sets used to create tuples

- The keywords used to denote membership or non-membership are YES and NO

Syntax

```
set
C  "crops"                /wheat,maize,tomato/
T  "techniques"           /T0*T2/
CT(C,T) "feasible combinations crop-technique"
;
  CT(c,t) = yes;
  CT('maize','t0') = no;
  CT('tomato','t0') = no;
```

Conditionals

Conditionals (subsets)

Used to define parameters, variables or equations only for selected set elements

Example

```
set
C 'crops'  /wheat,maize,tomato,potato
            sunflower,soya,sugarbeet/

cereals(c) 'cereals' /wheat,maize/
;
parameter SB(c) 'subsidy' ;

SB(cereals) = 150 ;

display SB;
```

Conditionals ($ operator)

Used in conditional assignments, expressions and equations

The condition: if (x > 3), then y = 2

can be modelled in GAMS as follows:

Syntax

```
y$(x>3) = 2;
*"y, such as x is greater than 3, equals 2"
```

Conditionals ($ operator)

Conditional assignments

- **$ on the left**: no assignment is made unless the logical condition is satisfied
- **$ on the right**: an assignment is always made (the term will be zero when the condition is not satisfied)

Conditional equations

- Dollar operator within the algebra (analogous to $ on the right)
- Dollar control over the domain definition (analogous to $ on the left)

Let us look at some examples !

$ operator (example)

dollar_conditions.gms

```
*--- dollar on the left  "p_left, such that a=a1, equals 50"

p_left(a)$(ord(a)=1) = 50 ;

*--- dollar on the right  "if a=a1, then p_right=50, else p_right=0"

p_right(a) = 50$(ord(a)=1) ;
```

Non-linear models (initial values and bounds)

Working with non-linear models

Assigning initial values to variables

- They help GAMS to find the optimal solution and speed up the iteration process (default value is zero)
- They need to be entered before the solve statement

Syntax

```
var_name.L = ini_value ;
solve nlpModel ...
```

Working with non-linear models

Providing lower and upper bounds

- They speed up the iteration process
- They are useful when working with variables that are undefined if another variable becomes zero
- They need to be entered before the solve statement

Syntax

```
var_name.LO = lower_value ;
var_name.UP = upper_value ;
solve nlpModel ...
```

Fixed variables

Variables can be endogenous or exogenous depending on the model run

Fixed variables

- It is possible to fix the value of a variable through the suffix **.fx** => equivalent to setting lower and upper bounds equal to the fixed value

example

```
variable v(r,p,m,y) 'endogenous variables' ;

* fixed value for ad-valorem tariff
v.fx(r,p,"TAV",y) = 0 ;
```

Display options

GAMS features (DISPLAY)

DISPLAY statement:

- Instruction that allows us to choose which elements we want to display in the output file (**.lst**)
- We can display data, model results or calculations with data or results.
- When displaying parameters, we do not add the domain of definition

Syntax

```
display price, cost;
```

GAMS features (DISPLAY)

DISPLAY statement:

- When displaying model results we have to use the DISPLAY command after the SOLVE statement.
- Four values are associated to every variable and equation in the model. Hence, when displaying variables and equations, we need to specify which value we want to display.

Syntax

```
display Z.L;

display QD.L, PD.L;
```

GAMS features (OPTION)

Option display

- To modify the display formatting

Syntax

```
option parName:decimals:rowItems:colItems;

* example
option result:1:1:1;
```

GAMS features (OPTION)

Option decimals

- Specifies the default number of decimal places to be printed by all subsequent display statements
- The default value is 3 and the range is from 0 to 8

Syntax

```
option decimals=number;

* example
option decimals=1;
```

Comments

Three ways to include comments

1. To start a line with an asterisk (*) in the first position (single line comments). GAMS will ignore this line.

Syntax

```
* this line contains explanatory text

* this is a comment
but * this is not a comment
```

Three ways to include comments

2. To use $ontext-$offtext delimiters (multiple line comments). GAMS will ignore the text between delimiters.

Syntax

```
$ontext

this section contains
explanatory text

$offtext
```

Three ways to include comments

3. To use the options $eolcom (end of line comment) or $inlinecom (inside line comment).

Syntax

```
$eolcom #
$inlinecom {}

X = 1 ;  # this is a comment
Y = 2 ;  {this is also a comment} Z = 3 ;
```

GDX utilities

GDX utilities (GDX viewer)

GDX (GAMS data exchange) files:

- Files that store the values of one or more GAMS symbols (sets, parameters, variables and equations)
- Intermediary files (between GAMS language and another software package)
- Binary files that are portable between different platforms. They can be used:
  - To prepare data for a GAMS model
  - To pass results of a GAMS model into different programs
- In GAMSIDE, can be handled using the GDX-viewer

GDX utilities (GDX viewer)

Storing parameters in GDX format

- During execution of a GAMS model we can write to GDX files using the execute_unload command
- If no path is specified, the gdx file will be written in the current project directory

Syntax

```
execute_unload 'file_name.gdx' parameter_name;

execute_unload 'file_name.gdx' ;
```

GDX utilities (compilation time)

During compilation of a GAMS model, we can read data from a GDX file into GAMS:

Instruction	Description
$GDXIN file_name	Specify the GDX file to be used for reading
$GDXIN	Close the current GDX input file
$LOAD S1 S2	Read GAMS symbols S1, S2
$LOAD S1=gdx1	Read GAMS symbol S1 with corresponding name gdx1

GDX utilities (compilation time)

During compilation of a GAMS model we can write to a GDX file:

Instruction	Description
$GDXOUT file_name	Specify the GDX file to be used for writing
$GDXOUT	Close the current GDX output file
$UNLOAD S1 S2	Write GAMS symbols S1, S2
$UNLOAD S1=gdx1	Write GAMS symbol S1 with corresponding name gdx1

GDX utilities (execution time)

During execution of a GAMS model we can read and write GDX files with the following statements:

Syntax

```
* to read from GDX

execute_load 'file_name.gdx' par1, par2=P2;

* to write to GDX

execute_unload 'file_name.gdx' par3, par4=P4;
```

GDX utilities (GAMS-EXCEL link)

GDXXRW utility: allows reading from (and writing to) an Excel spreadsheet

Syntax

```
* importing data from EXCEL

$CALL "GDXXRW.EXE excel_file1.xlsX index=sheet1!A3";

$CALL "GDXXRW.EXE file2.xlsX par=P2 rng=sheet2!A3 rdim=1 cdim=1";

* exporting data from EXCEL

execute "GDXXRW.EXE gdx_file1.gdx index=sheet1!A3";

execute "GDXXRW.EXE file2.gdx par=P2 rng=sheet2!A3 rdim=1 cdim=1";
```

GDX2XLS

GDX2XLS is a tool to dump the complete content of a GDX file to an Excel spreadsheet (.xlsx or .xls file). Every identifier gets its own sheet in the Excel file.

Syntax

```
* Saving the file to GDX
execute_unload “sets_all.gdx"  ;
* exporting all sets to EXCEL
execute "gdx2xls sets_all.gdx" ;
```

GAMS functions

Set attributes

- Set elements have attributes that may be recovered during execution

Syntax

```
setname.attribute

where
 setname is the name of the set
 attribute is one of the following
     ord      uel
     pos      val
     off      len
```

Set attributes (example)

set_attributes.gms

```
p_1('ord',a) = a.ord ;
p_1('pos',a) = a.pos ;
p_1('off',a) = a.off ;
p_1('uel',a) = a.uel ;
p_1('val',a) = a.val ;
p_1('len',a) = a.len ;
```

GAMS functions

Common mathematical functions

sum	sum of set indexed expressions
prod	product of set indexed expressions
sqr	square of an expression or term
sqrt	square root of an expression or term
log	natural logarithm
abs	absolute value
max, min	maximum or minimum of a set of expressions or terms
smax, smin	maximum or minimum of set indexed expressions or terms

GAMS functions

Basic statistical functions

normal(MEAN,STDDEV)	generates a random number with normal distribution with mean MEAN and standard deviation STDDEV
uniform(LOW,HIGH)	generates a random number between LOW and HIGH with uniform distribution
uniformInt(LOW,HIGH)	generates an integer random number between LOW and HIGH with uniform distribution

GAMS features (random numbers)

- Some GAMS functions generate random numbers following a specified probability distribution

Syntax

```
set N  "number of draws"  /n01*n40/ ;

parameter mean,sigma,p_normal,low,high,p_uniform ;


p_normal(n)  = normal(mean,sigma);

p_uniform(n) = uniform(low,high);
```

Normal(mean,sigma) generates random numbers with normal distribution

Uniform(low,high) generates random numbers between LOW and HIGH with uniform distribution

Functions (example)

gams_functions.gms

```
* NORMAL     random number normally distributed
(mean,sigma)
STAT(s,'norm(0,1)') = normal(0,1);
STAT(s,'norm(5,2)') = normal(5,2);

* UNIFORM    random number with uniform distribution
between x and y
STAT(s,'unif(0,1)')   = uniform(0,1);
STAT(s,'unif(10,60)') = uniform(10,60);
```

Control variables

Control variables

Control variables are used for conditional compilation

$setglobal is used to define a global control variable (available throughout the code)

Global variables are destroyed using $dropglobal

Syntax

```
$setglobal varname varvalue

      where varname is the name of the variable
            varvalue can contain text or a number

$dropglobal varname
```

Control variables

$setlocal is used to define a local variable (accessible only in the code module where defined)

Local variables are destroyed using $droplocal

Syntax

```
$setlocal varname varvalue

      where varname is the name of the variable
            varvalue can contain text or a number

$droplocal varname
```

Control variables (example)

Commonly used to articulate complex conditions

control_variables.gms

```
$setglobal simc  IND

v.fx("%simc%",'BA',"TAV",y)
  = v.l("%simc%",'BA',"TAV",y) + 0.1 ;
```

Control variables (example)

Commonly used to articulate complex conditions

control_variables.gms

```
$setglobal simc  swhe

parameter ygrowth(cact) 'yield growth';
ygrowth(cact) = 0.05;
ygrowth("%simc%") = 0.10;
```

Control variables for paths

- Control variables can also be used to indicate the relative path as in this example
- Create control variables to define the paths (to data directory, results directory and scenario directory) and use the control variables throughout the code

Example

```
$setglobal datadir  ..\data
$setglobal resdir  ..\results
$setglobal scendir  ..\scen
```

Conditional compilation

$include

$include inserts in an input file the content of an external file (data or GAMS statements)

Syntax

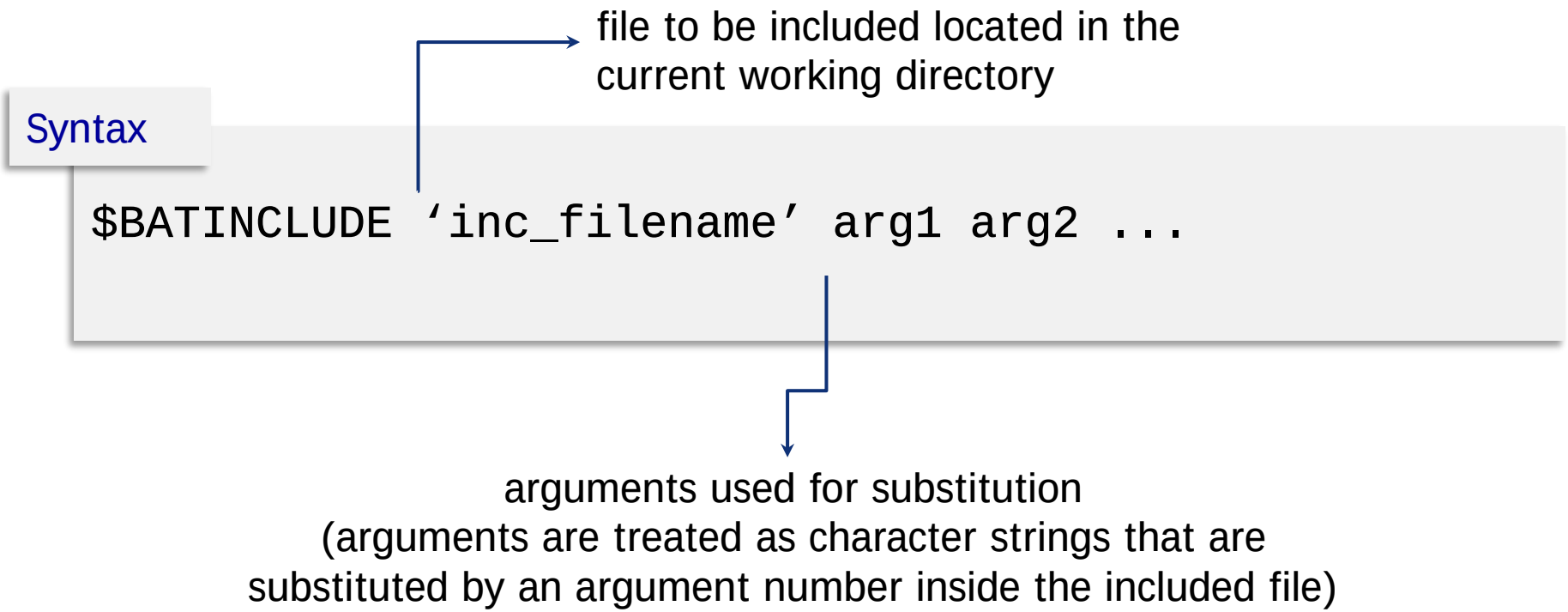
```
$INCLUDE 'inc_filename'

$INCLUDE 'C:\inc_file_path\inc_filename'
```

file to be included located in the current working directory

when the file to be included is not located in the current working directory, the path has to be specified

$batinclude

$batinclude inserts in an input file the content of an external file and it also passes on arguments)



if / else / elseif

If, else, elseif: logical conditions

Syntax

```
If (logical condition,
    statements to be executed if true ;);

If (logical condition,
    statements executed if condition true;
else
    statements executed if condition not true;);

If (logical condition,
    statements to be executed if true ;
Elseif logical condition,
    statements executed if this conditional is true
and the earlier one is false );
```

if / else / elseif (example)

Abort: causes the job to stop with an execution error and displays information

conditionals_if_elseif.gms

```
parameter P1; P1=-3;
if(P1 < 0,abort "stopped because P1 < 0",P1;);
parameter P2; P2=3;
display P2;
```

if / else / elseif (example)

conditionals_if_elseif.gms

```
set a  /a1*a3/ ;
parameter P3; P3(a)  = ord(a);
*   ---- reassign values
loop(a,
   if ( sameas(a,"a1"), P3(a) = 50;
      else if ( sameas(a,"a2"), P3(a) = 75;
         else if ( sameas(a,"a3"), P3(a) = 100;
                  )
               )
        );
     );
display P3;
```

$if / $ifi statements

Execution of a GAMS statement when a conditional is true

Syntax

```
$if conditional statement_to_execute
$ifi conditional statement_to_execute
or
$if conditional
  statement_to_execute
```

- $if is case sensitive ($ifi is a case insensitive variant)
- The conditional is evaluated at compile time, so does not involve GAMS calculated numbers

$if / $ifi statements (example)

conditionals_if_ifthen.gms

```
set a /a1*a3/ ;
parameter p1;

$ifi not defined a display "a is not defined";
$ifi not declared b display "b is not defined";
$ifi not defined p1 display "p1 is not defined";
```

$if / $ifi statements (example)

conditionals_if_ifthen.gms

```
set a /a1*a3/ ;
parameter p1;

$ifi not defined a display "a is not defined";
$ifi not declared b display "b is not defined";
$ifi not defined p1 display "p1 is not defined";
```

$if / $ifi statements (example)

Abort: causes the job to stop with an execution error and displays information

conditionals_if_elseif.gms

```
* Condition to abort the model in case the base data
is missing

$ifi not exist "base_data.gdx" $ABORT "base_data.gdx"
is missing, in %system.fn%, line %system.incline%
```

$ifthen / $iftheni statements

Execution of a GAMS statement when a conditional is true

Syntax

```
$iftheni
conditional statement_to_execute
conditional statement_to_execute
$endif
```

- $ifthen is case sensitive ($iftheni is a case insensitive variant)
- The conditional is evaluated at compile time, so does not involve GAMS calculated numbers

Model attributes and options

Model attributes and options

- Options controlling the content of the LST file
  - $ options
  - Option statements
- Options controlling the solver
  - Model options
  - Model attributes
- https://www.gams.com/latest/docs/userguides/userguide/_u_g__dollar_control_options.html

GAMS language (model attributes)

Sufix	Description	Sufix	Description
modelstat	Model status	solvestat	Solver status
1	Optimal	1	Normal completion
2	Locally optimal	2	Iteration interrupt
3	Unbounded	3	Resource interrupt
4	Infeasible	4	Terminated by solver
5	Locally infeasible	5	Evaluation error limit
6	Intermediate infeasible	6	Unknown
7	Intermediate non-optimal	7	(unused)
8	Integer solution	8	Error preprocessor error
9	Intermediate non-integer	9	Error setup failure
10	Integer infeasible	10	Error solver failure
11	(unused)	11	Error internal solver error
12	Error unknown	12	Error post-processor error
13	Error no solution	13	Error system failure

GAMS language (model attributes)

Attributes that can be controlled by the user

Sufix	Description	Default	Global option
iterlim	Iteration limit	1000	iterlim
limcol	Number of columns displayed for each block of variables	3	limcol
limrow	Number of rows displayed for each block of equations	3	Limrow
reslim	Time limit for solver (CPU seconds)	1000	Reslim
optfile	Option file usage	0	
solprint	Solution print option	1	solprint

Model options (example)

example

```
$offlisting
option limrow=18,limcol=0,solprint=off;

option nlp = conopt;

mod.solprint = 0;
mod.iterlim = 0;
mod.optfile = 0;
mod.limrow =0;
```

Setting environment variables

Environment variables

- GAMS recognizes the environment variable GDXCONVERT and GDXCOMPRESS which control the format with which GDX files are written.

0	do not compress gdx files (default)
1	compress gdx files

Syntax

```
$setenv GDXCOMPRESS number
```

Model development: partial solve

Save and restart

- Feature that allows for running the model in pieces (intermediate work is saved at the end of each run)
- Useful for:
  - Separation of model and data
  - Model development: by splitting the model in pieces, we can run only the modified ones.
  - Running multiple scenarios: This feature can save time when running scenarios and managing results.

Save and restart

Syntax

```
Gams piece1.gms s=save1
Gams piece2.gms r=save1 s=save2
Gams piece3.gms r=save2
```

Example

```
Gams dataHandling.gms    s=s1
Gams runModel.gms        r=s1 s=s2
Gams resultReporting.gms r=s2
```