

# Introduction to GAMS

G Cornelis van Kooten


# GAMS

- Powerful software for solving LP, NLP, MLP and CGE models.
- Easy to use when setting up problems, especially in the case of very large problems: much easier than Excel.
- More powerful than Excel, Matlab, Maple, Mathematica, etc. for solving very large, non-linear programs
- Enables researcher to access a variety of powerful solvers using the same program. Solvers include Minos, CPLEX, ConOpt, Lindo, XA, many other commercial solvers

# GAMSIDE

- **GAMS**: General Algebraic Modeling System  
plus
- **IDE**: Integrated Development Environment  
A Windows graphical interface to run GAMS

# Summary steps for using GAMS

1. Install GAMS and the IDE on your computer making an icon
2. Open the IDE through the icon
3. Create a project by going to the  selection in the upper left corner.

What is a project? The GAMSIDE employs a “Project” file (1) to identify a location where all saved files are to be placed (to place files elsewhere use the save as dialogue) and where GAMS looks for files when executing, and (2) to which project files and program options are saved. It is recommended that you define a new project every time you wish to change the file storage directory.

## Summary steps for using GAMS

4. Define a project name and location. Put it in a directory you want to use. Choose a suitable subdirectory and create a subdirectory with the name of the project. In turn, call your project by that name so that it is called *project.gpr*, where gpr stands for GAMS project.
5. Create or open an existing file of GAMS instructions
6. Prepare the file so you think it is ready for execution
7. Run the file with GAMS by clicking on the run button
8. Open and navigate around the output

# A GAMS Program

## INPUTS:

- SETS
  - declaration
  - assignment of members
- DATA (Parameters, Tables, Scalars)
  - declaration
  - assignment of values

# A GAMS program (cont)

- VARIABLES
  - declaration
  - assignment of type
- Assignment of bounds and/or initial values (OPTIONAL)

# A GAMS program (cont)

- EQUATIONS
  - declaration
  - definition
- MODEL and SOLVE Statements
- DISPLAY Statement (OPTIONAL)



# Equations in GAMS

Two types of equations in GAMS:

1. Define parameters
2. Model equations (may be inequalities)

**Not indexed:** Objective is a single value

$$Z = \text{sum}((i,j), a(i,j)*x(i,j));$$

**Indexed:** Examples

$$R(i) = \text{sum}(j, c(j)*x(i,j));$$

$$R(\text{item}) = \text{sum}(\text{city}, c(\text{city})*x(\text{item},\text{city}));$$

# Example problem: Minimize transportation costs

$$\min \sum_i \sum_j c_{ij} x_{ij} \text{ (\$)}$$

$$s.t. \sum_j x_{ij} \leq a_i \quad (\text{supply limit at plant } i)$$

$$\sum_i x_{ij} \geq b_j \quad (\text{satisfy demand at market } j)$$

# SETS

i plants /Seattle, San-Diego/;

j market /NY, Chicago, Kancity/;

# PARAMETERS

a(i) capacity at plant i

/ Seattle 350

San-Diego 600 / ;

b(j) demand at market j

/ NY 325

Chicago 300

Kancity 275 / ;

TABLE  $d(i, j)$  distance in '000s miles;

	NY	Chicago	Kancity
Seattle	2.5	1.7	1.8
San-Diego	2.5	1.8	1.4

SCALAR  $f$  freight cost in \$ per '000s miles /8/ ;

PARAMETER  $c(i, j)$  transport cost in k\$ per unit ;

$$c(i, j) = f * d(i, j) / 1000;$$

## VARIABLES

$x(i, j)$  shipment quantities

$z$  total costs in k\$;

Positive variable  $x$ ;

## EQUATIONS

supply(*i*)      supply limit at plant *i*  
demand(*j*)      satisfy demand at market *j*  
cost              objective function is to minimize cost  
;

cost..             $z = E = \text{sum}((i,j), c(i,j)*x(i,j));$   
supply(*i*)..       $\text{sum}(j, x(i,j)) = L = a(i);$   
demand(*j*)..       $\text{sum}(i, x(i,j)) = G = b(j);$

**MODEL** transport /all/;

**SOLVE** transport using LP minimizing z;

**DISPLAY** x.l, x.m ;

# Specify upper and lower bounds in lieu of constraints

$x.\text{up}(j) = 50;$

Same as an  $x_j \leq 50$  constraint

$y.\text{lo}(\text{"2"})=20; \rightarrow y_2 \geq 20$  constraint

(Note reference to index 2)

$\text{cows}.\text{up}(\text{"3"}) = 200 + \frac{1}{2} x.\text{up}(\text{"3"});$

$\rightarrow \text{cows}_3 \leq 200 + \frac{1}{2} (50)$  constraint

# Report Writing in GAMS

DISPLAY can be used after SOLVE statement in GAMS to print out calculation results:

- 1) variable name followed by .L → value of solution for that variable
- 2) variable name followed by .M → reduced cost associated with that variable
- 3) equation name followed by .L → left-hand side value of equation
- 4) equation name followed by .M → shadow price (dual variable) associated with equation

Dual slack variables = RHS of equation – associated .L value

Example displays level and marginal values of X and shadow price of CON1:

```
DISPLAY X.L, X.M, CON1.M;
```

# PUT option and Report Writing

Define output file name and identify where output is to be written. If no path is given, default is project subdirectory and file is file name with .put extension

file Output /D:\work\output.txt/ ;

\*Set page width

Output.pw=5000;

\*Number of decimal places in output file

Output.nd=0;

Solve *modelname* using LP maximizing z;

Now comes the writing component. A comma is needed to create a comma separated value file, and a / is used to indicate a carriage return.

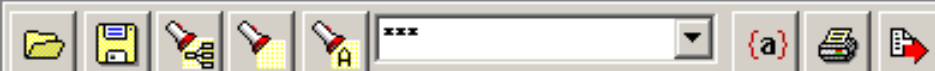


## PUT option (cont)

```
put Output;  
  put "Solver status ", modelname.solvestat/;  
  put "Model status ", modelname.modelstat/;  
  put "Objective value ", z /;  
  put "Period" ", " "Price" ", " "Quantity" ", "/;  
  loop(t, put t.tl ", " price.l(t) ", " quan.l(t) ", "/);  
  put/;  
putclose Output;
```

Note how the index  $t$  needs to be identified in the loop with a `.tl` as opposed to only `.l`.

An example of how to write output is provided in the next two slides for an electricity grid model.



Alberta\_model2.gms

```
*****  
** Output writing section  
*****  
  
emissions.l(h)=sum(p,total.l(p,h)*emfactor(p)/maxefficiency(p)/0.2778);  
  
cost.l(p,h)=fuelcost.l(p,h)+OMcost(p);  
  
peakcosttotal.l=(sum(h,cost.l('peak',h))+ peakcap.l*fixedcost('peak'))/1e6;  
  
total_all.l(h)=sum(p,total.l(p,h));  
  
file result / result.txt /;  
result.pw=10000;  
put result 'Alberta Electricity Grid Model' ///;  
put result 'Run on ' system.date ' using source file ' system.ifile ///;  
put "Model status ",energy.modelstat/;  
put "Solver status ",energy.solvestat/;  
put "Objective ," z.l / "Peak capacity ," peakcap.l /  
    "Fixed cost of wind," fixedcost('wind')/  
    "Wind capacity," capacity('wind') ///;  
put ",," "POWER OUTPUT BY SOURCE (MW)" ",," ",," ",," ",," ",," ",," ",,"  
    "COST - $ PER MW" ",," ",," ",," ",," "UPPER RESERVOIR"  
    ",," ",," ",," "LOWER RESERVOIR" ",," /  
put "Hour" ",," "Total" ",," "Coal" ",," "Gas" ",," "Peak" ",," "Wind" ",,"
```

252: 1

Insert



Alberta\_model2.gms

```
put "Objective ," z.l / "Peak capacity ," peakcap.l /
    "Fixed cost of wind," fixedcost('wind')/
    "Wind capacity," capacity('wind') ///
put ",," "POWER OUTPUT BY SOURCE (MW)" ",," ",," ",," ",," ",," ",," ",," ",,"
    "COST - $ PER MW" ",," ",," ",," ",," ",," "UPPER RESERVOIR"
    ",," ",," ",," "LOWER RESERVOIR" ",," /
put "Hour" ",," "Total" ",," "Coal" ",," "Gas" ",," "Peak" ",," "Wind" ",,"
    "Hydro" ",," "Demand/load" ",," "Coal" ",," "Gas" ",," "Peak" ",,"
    "Wind" ",," "Hydro" ",," "Volume" ",," "Outflow" ",,"
    "Inflow" ",," "CO2 Output" ",," "PumpFlow" ",,"
    "Volume" ",," /;
loop (h, put h.tl ",," total_all.l(h) ",," total.l('coal',h) ",,"
total.l('gas',h) ",," total.l('peak',h) ",,"
total.l('wind',h) ",," total.l('hydro',h) ",," demand(h)
",," cost.l('coal',h) ",," cost.l('gas',h) ",,"
cost.l('peak',h) ",," cost.l('wind',h) ",,"
cost.l('hydro',h) ",," volume.l(h) ",," outflow.l(h) ",,"
inflow(h) ",," emissions.l(h) ",," reflow.l(h) ",,"
volume2.l(h) /;);
put /;
putclose;
```

252: 1

Insert

# Calling external Files from within GAMS

- \$Include *externalfilename*

where *externalfilename* can include full path or just file name if located in the current working directory as declared in GAMS project.

The file to include consists of GAMS code.

- The ‘master’ GAMS file treats the statements in the include file as a continuation of the program code precisely at the \$Include point.
- Other commands for including external files are: \$Batinclude, \$Libinclude, \$Sysinclude
- \$Oninclude and \$Offinclude turn the echo print of the include file in the LST file on and off (default is ‘on’)

# Example from grid model

**PARAMETER** inflow(h) hourly inflow of water m<sup>3</sup> per s  
\$Include inflow

**PARAMETER** demand(h) hourly demand for electricity  
\$Include demand2003

**PARAMETER** wpower(h) hourly wind power (MW)  
\$Include wpowers

Data are found in **inflow.gms**, **demand2003.gms** and **wpowers.gms**

# Comma Separated Value (CSV) Files

- Spreadsheets and other programs (e.g., Matlab) can read and write CSV files
  - Commas separate fields
  - Text items can be in quotes
- \$ondelim – tells GAMS that entries which follow are in CSV format
- \$offdelim – tells GAMS to ‘turn off’ use of CSV format

# Examples using CSV format

## SETS

K index k /K1\*K3/

J index j /J1, J2, J3/;

TABLE CSVEX1(K,J) Data in csv Format

\$ondelim

dummy, J1, J2, J3

K1, 16, 34, 12

K2, 3, 11, 5

K3, 52, 16, 4

\$Offdelim

Note use of 'dummy' as the 'placeholder' in a table (see also below)

Alternatively an external file can contain  
the table:

SETS

K index k /K1\*K3/

J index j /J1\*J3/;

TABLE CSVEX2(K,J) data in csv format

\$ondelim

\$Include externalcsvdatafile

\$offdelim



The file *externalcsvdatafile.gms* contains:

dummy, J1, J2, J3

K1, 16, 34, 12

K2, 3, 11, 5

K3, 52, 16, 4

Note: ‘dummy’ is needed in the csv format to use up the space over the set elements defining the table rows – in the first row, first column.

# GAMS Model

\$Title McCarl & Spreen Chapter 3 Example

## SETS

i index 1 /1\*3/

j index 2 /1, 2/

;

**PARAMETER** b(j) rhs constraint values

/ 1 10

2 3 /

;

**TABLE** a(j,i) Table of technical constraints

1 2 3

1 1 1 1

2 1 -1 0

;

**PARAMETER**  $c(i)$  coefficients in obj

/1 3

2 2

3 0.5/

;

**VARIABLES**

z objective value

$x(i)$  activities

;

**EQUATIONS**

OBJ Objective function

CON(j) Constraint

;

\* -----  
\* Construction of the actual LP model  
\* -----

OBJ ..  $z = E = \text{sum}(i, c(i)*x(i));$

CON(j)..  $\text{sum}(i, a(j,i)*x(i)) =L= b(j);$

**MODEL** McSpreen /all/;

$x.lo('1')=0; x.lo('2')=0;$

**SOLVE** McSpreen using LP maximizing z;

# Solution to previous GAMS problem

SinkCapacity

McSreen\_Chap3\_example.gms    McSreen\_Chap3\_example.lst

- Compilation
- Equation Listing
- Equation
- Column Listing
- Column
- Model Statistics
- Solution Report
- SolEQU
- SolVAR

```

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

MODEL      McSreen      OBJECTIVE   z
TYPE       LP           DIRECTION  MAXIMIZE
SOLVER     CPLEX       FROM LINE  48

**** SOLVER STATUS      1 NORMAL COMPLETION
**** MODEL STATUS       3 UNBOUNDED
**** OBJECTIVE VALUE           26.5000

RESOURCE USAGE, LIMIT      0.015      1000.000
ITERATION COUNT, LIMIT     2          10000

GAMS/Cplex   Apr 21, 2006 WIN.CP.CP 22.2 031.034.041.VIS For Cplex 10.0
Cplex 10.0.1, GAMS Link 31

Presolve found the problem infeasible or unbounded.
Rerunning with presolve turned off.
Dual infeasible or unbounded. Switching to primal to aid diagnosis.
Model has an unbounded ray.

**** ERRORS/WARNINGS IN VARIABLE x(3)
      1 error(s): 1 Unbounded variable
    
```

SinkCapacity [a] [Print] [Run]

McScreen\_Chap3\_example.gms | McScreen\_Chap3\_example.lst

- Compilation
- Equation Listing
- Equation
- Column Listing
- Column
- Model Statistics
- Solution Report
- SolEQU
- SolVAR

```

**** ERRORS/WARNINGS IN VARIABLE x(3)
      1 error(s): 1 Unbounded variable

      LOWER      LEVEL      UPPER      MARGINAL
---- EQU OBJ
      .           .           .           1.000

      OBJ Objective function

---- EQU CON Constraint

      LOWER      LEVEL      UPPER      MARGINAL

1      -INF      10.000      10.000      2.500
2      -INF      3.000       3.000       0.500

      LOWER      LEVEL      UPPER      MARGINAL
---- VAR z
      -INF      26.500      +INF        .

      z objective value

---- VAR x activities
    
```

121: 1

- Compilation
- Equation Listing
- Equation
- Column Listing
- Column
- Model Statistics
- Solution Report
- SolEQU
- SolVAR

```

LOWER      LEVEL      UPPER      MARGINAL
---- VAR z          -INF      26.500      +INF      .
z objective value
---- VAR x activities
LOWER      LEVEL      UPPER      MARGINAL
1          .          6.500      +INF      .
2          .          3.500      +INF      .
3          -INF       .          +INF      -2.000 UNBND

**** REPORT SUMMARY :
0          NONOPT
0 INFEASIBLE
1 UNBOUNDED (UNBND)

EXECUTION TIME      =      0.000 SECONDS      2 Mb WIN222-145 Apr 21,

```

**Notice that GAMS provides a solution**



FarmSimPMP.gms RPGTradeModel.gms

```
* Whole Farm Insurance Model
* Following ideas in papeers by Calum Turvey, Keithe Coble and others
* Copyright G. Cornelis van Kooten and Esther Broere

SETS
    crop      crops /wheat,durum,oat,barley,flax,canola,pea,lentil,other,sf/
    iter      iterations in Monte Carlo Simulation /1*100/

alias(crop, r);

SCALARS delta, area, targetR, states;
$include scalars.gms

TABLE input(crop, *) Input values created in R file
$ondelim
$include input.csv
$offdelim
;

* ----- CONTENTS OF INPUT MATRIX CREATED IN R: -----
* cost - variable cost in $ per acre
* pr - mean price of the various crops in $ per tonne
* yld - average yield by crop in tonnes per acre
* obs - observed acres in each of the crops with total acres equal 1000

TABLE carlo(iter, crop) Monte Carlo revenue simulations
$ondelim
$include revenue.csv
$offdelim
;

* ----- Observed net revenue per acre by crop for historic data
PARAMETERS xrev(crop) 'net return per acre by crop (average over random states)'
            n1         for variance
```

1: 1

Insert







FarmSimPMP.gms RPGTradeModel.gms

```
* ----- Observed net revenue per acre by crop for historic data
PARAMETERS xrev(crop) 'net return per acre by crop (average over random states)'
            n1         for variance
            n2         for skewness
;

xrev(crop) = sum(iter, carlo(iter,crop))/states;
n1 = 1/states;
n2 = states/((states-1)*(states-2));

* ----- CONSTRUCTION OF THE MODEL EQUATIONS
VARIABLES
    rev1         Objective for 1st stage of PMP
    rev2         Objective for 2nd stage of PMP
    V1           Variance value
    x(crop)      Optimal number of acres in each crop
    rev(iter)    Net revenue for each random state
    avrev        Average net revenue across all random states
    Skew         skewness value
    GR(iter)     Revenue for each random state w crop specific insurance
    Z(crop)      Insurance premium for each crop
;

POSITIVE VARIABLES x ;

EQUATIONS
    variance     minimize variance of yield
    revenu(iter) Net revenue for each iteration with optimal land use
    land         Total land use constraint
    average      Calculate average revenue across all random states
    obj1         Net revenue objective for 1st stage of PMP
    obj2         Net revenue objective for 2nd stage of PMP
    calib(crop)  Calibration constraints
```

1: 1

Insert





FarmSimPMP.gms RPGTradeModel.gms

```

        calib(crop)      Calibration constraints
;

variance.. V1 =E= n1*sum(iter, sqr(0.001*(rev(iter)-avrev)));

revenu(iter)..  rev(iter) =E= sum(crop, carlo(iter,crop)*x(crop));
average..      avrev =E= sum(iter, rev(iter))/states;

land..        sum(crop, x(crop)) =E= area;

obj1..        rev1 =E= sum(crop, x(crop)*(input(crop,'yld')*input(crop,'pr') -
                        input(crop,'cost')));

calib(crop)..  x(crop) =L= input(crop,'obs')+0.01;

MODEL pmp /obj1, land, calib/

OPTION QCP=CPLEX;

SOLVE pmp using LP maximizing rev1;

* Assume TC = m0 x + 0.5 m1 x^2
PARAMETERS m1(crop)  Slope parameter for MC curve
            m0(crop)  Intercept parameter for MC curve
;

m1(crop) = 2*calib.m(crop)/input(crop,'obs');
m0(crop) = input(crop,'cost') - 0.5*m1(crop)*input(crop,'obs');

EQUATIONS
    target          Revenue target
    GrossRev(iter)  Gross revenue insurance w crop specific revenue protect
;

```

1: 1

Insert





FarmSimPMP.gms RPGTradeModel.gms

```
obj2..    rev2 =E= sum(crop, input(crop,'pr')*input(crop,'yld')*x(crop)
           - m0(crop)*x(crop) - 0.5*m1(crop)*x(crop)*x(crop));

target..  rev2 =G= targetR;

GrossRev(iter).. GR(iter) =E= rev(iter)
                + sum(crop, max((Z(crop)-rev(iter)),0)*x(crop))
                - delta*sum(crop, max((Z(crop)-xrev(crop)),0)*x(crop));

MODEL baseRev /obj2, land/;

MODEL baseVar /variance, obj2, revenu, average, target, land/

MODEL GrossRevIns /variance, obj2, revenu, GrossRev, average, target, land/;

*OPTION QCP=MINOS5;
OPTION NLP=CONOPT;
*OPTION NLP=MINOS5;
OPTION DNLP=SNOPT;
*OPTION DNLP=CONOPT;
*OPTION DNLP=MINOS5;

* -----Minimize Variance s.t. revenue target-----
SOLVE baseRev using QCP maximizing rev2;
*
* Write solution
*
file RevOut /REV.csv/;
put RevOut;
  put rev2.l /
  put 0.0 /
*  put V1.l /
  loop (crop, put x.l(crop) /;
```

1: 1

Insert





FarmSimPMP.gms RPGTradeModel.gms


```
SOLVE baseRev using QCP maximizing rev2;
*
* Write solution
*
file RevOut /REV.csv/;
put RevOut;
  put rev2.1 /
  put 0.0 /
*   put V1.1 /
  loop (crop, put x.1(crop) /;
);
putclose RevOut;

SOLVE baseVar using NLP minimizing V1;
*
* Write solution
*
file VarOut /VAR.csv/;
put VarOut;
  put rev2.1 /
  put V1.1 /
  loop (crop, put x.1(crop) /;
);
putclose VarOut;

* -----
* ----- Gross Revenue insurance minimizing variance -----
SOLVE GrossRevIns using DNLP minimizing V1;
*
* Write solution
*
file GRIPout /GRIP.csv/;
put GRIPout;
  put rev2.1 /
```

1: 1

Insert

 IntroGAMS.pptx - Microsoft PowerPoint






FarmSimPMP.gms    RPGTradeModel.gms

```

put rev2.1 /
put V1.1 /
loop (crop, put x.1(crop) /;
);
putclose GRIPout;
* -----
* -----
* Semi-variance model:
* ----- Base case as above -----
PARAMETERS
    standev    standard deviation
;
standev = sqrt(V1.1);
EQUATIONS
    skewness    maximize skewness
;
skewness.. Skew =E= n2*sum(iter, power(((rev(iter)-avrev))/standev,3));
MODEL semivariance /variance, obj2, skewness, revenu, average, target, land/;
SOLVE semivariance using NLP maximizing Skew;
*
* Write solution
*
file SkewOut /SEMI.csv/;
put SkewOut;
    put rev2.1 /
    put V1.1 /
    loop (crop, put x.1(crop) /;
);
putclose SkewOut;
* -----
    
```

1: 1    Insert