Software Package for Doing Applied (Computational) Economic Modeling

Wishlist of Economist (Students/Professionals)

1) Syntax should be self-evident.

⇒ GAMS is intuitive because it is very close to the standard mathematical representation of (economic) models and uses the standard terminology.

2) Language should be compact to avoid „repetitive“ typing.

⇒ GAMS uses (algebraic) sets and detached-coefficient matrix notation which accommodates an efficient representation of large dimensional models using only a few lines of code (vector syntax).

3) Separation of logic and data - it should be easy to develop a small toy model from scratch and then blow it up to larger dimensions.

⇒ GAMS allows to keep data separate. Once correctly specified in logical terms a problem can be easily scaled up to higher dimensions (vector/set syntax).

4) „I don’t want to take care on solution algorithms at all!!“

⇒ GAMS accommodates the separation of model formulation and model solution GAMS passes on the problem definition to separate solver programs which return solution values back to the GAMS program.

5) Flexible Interface to other „Systems“ (spreadsheet, graphical plotting, ...)

⇒ GAMS provides an open environment to communicate with other systems. There are various ready-made tools (GAMS tailored macros) available for doing:

- spreadsheet imports/exports
- plotting
- sensitivity analysis (batch programming)
- interactive modeling (web)

Basic Components for the Algebraic Specification of Economic Problems:

1) Definition of Problem Dimensionality:

⇒ The dimensions of the problem are specified by sets which summarize multiple entities of the same type and denote the domain of decision variables.

⇒ Algebraically these sets are controlled by set indices.

⇒ Example for a set in the Transmission-model: power plants

2) Declaration of Decision Variables and Data Items (Parameters):

⇒ Declaration of decision variables and parameters is indexed over associated sets to provide a compact problem description.

⇒ Examples in the Transmission-model:

- decision variables: transmitted electricity.
- parameters: demand in a region, production limits

3) Specification of Functional Relationships:

⇒ Relationships between the various decision variables are described by means of equations (inequalities).

⇒ Example in the Transmission-model: transmission possibilities (cost function, demand and supply balance).

Table 2: Basic Components of a GAMS model

GAMS components:	Described in :
Sets	Section 2.1
Declaration	
Assignment of set members	
Parameters (Data)	Section 2.2
Declaration	
Assignment of values	
Variables	Section 2.3
Declaration	
Assignment of type	
Equations	
Declaration	Section 3.1
Definition	Section 3.2
Running a Model	
Model definition	Section 4.1
Assignment of bounds / initial values to variables	Section 4.2
Display statement	Section 4.3
Solve statement	Section 4.4
Invoking a GAMS run	Section 4.5
GAMS Output and Solution Report	
Compilation output and compilation errors	Section 5.1
Execution output and execution errors	Section 5.2
Solve output and solve errors	Section 5.3

Some general rules for GAMS programming:

1) The creation of GAMS entities involves always two steps: declaration and assignment.

⇒ declaration: indicate the existence of the entity type and give it a name (naming convention: name must start with a letter which then can be followed by up to nine more letters or digits).

⇒ assignment: specific values (e.g. data assignment) or forms (e.g. function specification)

⇒ documentation: GAMS allows the user to include explanatory text as part of the declaration of any entity.

2) Unique naming: A GAMS entity may not be declared more than once..

3) GAMS syntax rules are quite relaxed.

⇒ Users are e.g. free to place blank spaces and lines upon their needs.

⇒ GAMS allows multiple lines per statement and vice versa.

⇒ GAMS is not case sensitive.

4) Semicolons should always be used to indicate the end of a GAMS statement.

Specifying Key Entities of a Model: Sets, Parameters, Decision Variables

1) Defining the model dimensions (structure): SETS

⇒ sets equivalent to indices in algebraic notation

⇒ sets define model's dimension (in terms of their domain, that is the number of set elements)

⇒ set elements are treated as strings

2) Declaration of data entities and data assignment: PARAMETERS

⇒ joint declaration and data assignment

⇒ separate declaration and data assignment via algebraic expressions („=")

⇒ one-dimensional parameters → SCALARS

⇒ parameter data in list form → TABLES

⇒ reference to specific elements in quotes

⇒ multi-dimensional parameters

3) Declaration of endogenous (decision) variables: VARIABLES

⇒ sign conditions

⇒ GAMS optimization software for mathematical programming problems:

- formal set-up of mathematical programs is required (objective function subject to side constraints)
- formal need for free variable (lhs of objective function) even though there is nothing to optimize (DUMMY variable)

Specification of Functional Relationships: EQUATIONS

1) Declaration:

   ⇒ name of equation (incl. domain), optional documentary text

2) Definition:

   ⇒ name of the equation including its domain followed by the symbol '..'

   ⇒ algebraic relationship composed of a

   - left-hand-side expression,
   - the indication of the form of the equation by means of a relational operator, i.e.:

     =L= less than or equal to,

     =G= greater than or equal to, or

     =E= equal to
   - the right-hand-side expression.

   ⇒ Example for an indexed algebraic operation in our Transmission-model: summation statement SUM.

Final Steps to Run a Model

1) Formal model definition: MODEL statement

⇒ Example: MODEL Transmission /ALL/;

2) Setting bounds and initial values for variables

⇒ .LO,,.UP,.FX,,.L

⇒ numerical reasons

⇒ fixing of numeraire

3) Separation of model and solution: SOLVE statement

⇒ The SOLVE statement is composed of four element:

- It tells GAMS which model to solve (in our case: *Transmission*).
- It selects the solver to use (in our case: *LP* solver).
- It indicates the direction of the optimization (either *MINIMIZING* or *MAXIMIZING* - in the case of the dummy objective function we could use both) ,
- It refers to the objective variable (here: *z*).

⇒ Example: SOLVE Transmission USING LP MAXIMIZING z;

4) Writing Data: DISPLAY statement

⇒ write data into the listing file (data checks and user-designed solution reports)

⇒ applicable to sets, parameters, variables, and quoted text strings at any location of the program.

5) Invoking a GAMS Run From the System Prompt

⇒ Example: GAMS Transmission (under DOS)

GAMS Output and Solution Report

1) **Destination and Structure**

⇒ Default file: <program file name>.lst

⇒ Default listing file includes three major components:

- compilation output (compilation phase: check for syntax and consistency)
- the execution output (execution phase: execution of data transformations and model generation)
- the output produced by the solve statement (solution phase - outside GAMS)

2) Errors

⇒ indicated in the output report by four asterisks ****** at the beginning of a line in the output listing

⇒ a *$*-sign indicates the potential source of error directly below the offending GAMS expression followed by a numerical error code (explained at the end of the listing file)

3) Compilation Output and Compilation Errors

⇒ echo print, i.e. a copy of the original GAMS program

⇒ compilation errors (such as misspelling, ...)

⇒ maps (identifiers by type and location, identifiers in alphabetical order with comments)

4) **Execution Output and Execution Errors**

⇒ output associated with the DISPLAY statement.

⇒ execution errors from illegal data manipulations such as division by zero

5) Solve Output and Solve Errors

⇒ Equation Listing:

- individual constraints listed with their current values for set elements and parameters after data evaluation
- for nonlinear equations, the GAMS equation listing reports first-order Taylor approximations at the starting values of the variables

⇒ Column Listing:

- summary of the coefficients associated with each variable.

⇒ Model Statistics:

- size of the model (i.e. number of equations and variables),
- the degree of its non-linearity
- time requirements for generating and executing the model.

⇒ Solve Summary:

- SOLVER STATUS: normal termination or interruption due to limits (e.g. time limit, iteration limits) or internal difficulties.
- MODEL STATUS: type of solution (e.g.: optimal, locally optimal) or the type of problem associated with a solution failure (e.g. infeasible).

⇒ Solution Listing:

- results for the decision variables (level values, marginals) and equations

⇒ Solve errors:

- problems transforming the GAMS model in a solver compatible format
- problems faced by the solver when evaluating nonlinear functions or computing derivatives.

```
$Title  A Simple Market Model (LP-Formulation)
SETS
             i   power plants      / plant_01, plant_02 /
             j   regions markets   / reg_01, reg_02, reg_03 / ;
PARAMETERS

   a(i)      capacity of plant i in cases
         /   plant_01     350
             plant_02     600  /

             b(j)   demand at market j in cases
         /      reg_01           325
                reg_02           300
                reg_03           275  / ;

TABLE  d(i,j) distance in thousands of miles

                         reg_01      reg_02       reg_03
      plant_01             2.5          1.7          1.8
      plant_02             2.5          1.8          1.4  ;

SCALAR       f      transmission costs scalar per thousand miles /90/ ;

PARAMETER c(i,j)  transmission cost in thousands of dollars link ;

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

VARIABLES
       x(i,j)  transmission of electricity from plant to region
       z       total transmission costs  ;
POSITIVE VARIABLE x ;

EQUATIONS
       cost               define objective function
       supply(i)          observe supply limit at plant i
       demand(j)          satisfy demand at market j ;

cost..        z  =e=  sum((i,j), c(i,j)*x(i,j)) ;

supply(i)..   sum(j, x(i,j))  =l=  a(i) ;

demand(j)..   sum(i, x(i,j))  =g=  b(j) ;

MODEL transmission /all/ ;
SOLVE transmission using lp minimizing z ;

DISPLAY x.l, x.m ;
```