ISyE 6669: Deterministic Optimization

Introduction to GAMS

Shabbir Ahmed

Fall 2002

1 Introduction

The General Algebraic Modeling System (GAMS) is a high-level modeling system for mathematical programming problems. It consists of a language compiler and a stable of integrated high-performance solvers. GAMS allows the user to concentrate on modeling rather than the actual solution algorithms. The user builds up the model using the GAMS language and then calls one of the built-in solvers to solve the problem.

This article is meant to provide a rudimentary introduction to GAMS to get you started. For further information you are referred to the GAMS manual which can be downloaded from the GAMS website: `http://www.gams.com`. You should also consult Ron Rardin's Notes on GAMS available at:
`http://gilbreth.ecn.purdue.edu/~rardin/gams/notes.html`.

In the next Section, we describe the basic usage of GAMS through the portfolio optimization example discussed in class. Some important elements of the GAMS laguage are described in Section 3 and the GAMS output file is discussed in Section 4. The mathematical formulation and the corresponding GAMS input and output files for the example problem are provided in Appendices I, II and III respectively.

2 Basic Usage

The GAMS software is available on the Windows PCs and Unix Sun Workstations in the ISyE Graduate Lab.

2.1 Usage on Unix:

There are three basic steps to using GAMS for modeling and solving mathematical programs on Unix:

1. Use your favorite text editor (vi, pico, emacs etc.) to write a GAMS input file. This file would contain the description of the mathematical program and directions for solving and displaying results in the GAMS language (see Section 3). The GAMS input file should have a `.gms` extension. For example, the GAMS input file for the example problem in Appendix I is given in Appendix II. The example file is called `example.gms`.

2. Run the gams compiler on the input file. For example, to run the compiler on the file `example.gms`, you would enter the command:
`gams example.gms` <enter>
Running this command produces the following output:

```
GAMS Rev 121  Copyright (C) 1987-2001 GAMS Development. All
rights reserved Licensee: Shabbir
Ahmed                                  G001024:1053AE-SOL
          Georgia Tech, Industrial and Systems Engineering          DC3034
--- Starting compilation
--- example.gms(34) 1 Mb
--- Starting execution
--- Generating model example
--- example.gms(32) 2 Mb
---    3 rows, 4 columns, and 10 non-zeroes.
--- example.gms(32) 2 Mb
--- Executing MINOS5

 MINOS5        Mar 21, 2001 SOL.M5.M5 20.0 108.043.039.SOL

 Work space allocated           --    0.04 Mb
 Reading data...
 Reading nonlinear code...

   Itn  Nopt  Ninf  Sinf,Objective  Nobj  NSB
     1     1     1  9.22906820E+02

 Itn     1 -- Feasible solution.  Objective =   2.074207662E+04


 EXIT -- OPTIMAL SOLUTION FOUND

 Major, Minor itns              1           1
 Objective function       2.0742076618994E+04
 FUNOBJ, FUNCON calls           4           0
 Superbasics, Norm RG           0    0.00E+00
 Degenerate steps               0        0.00
 Norm X,      Norm PI    1.53E+03    1.83E+01
 Norm X,      Norm PI    7.75E+02    5.64E+02  (unscaled)
--- Restarting execution
--- example.gms(32) 0 Mb
--- Reading solution for model example
--- example.gms(34) 1 Mb
*** Status: Normal completion
```

3. Invoking the gams command first compiles the input file and calls a solver (depending upon the instructions). Once this is complete, the output is

written to a file with a `.lst` extension. This listing file can then be read by opening with a text editor. The listing or output file contains the solutions from the run (see Section 4).

2.2 Windows Usage:

For Windows, a convenient Integrated Development Environment (IDE) is available. Follow the following steps:

1. Locate the the `Gams` folder on the `Start` menu, and click on `gamside`. This should open up a window titled gamside.
2. On the `File` submenu, under `Project`, select `New Project` and provide the path where you want to store your files.
3. On the `File` submenu, click on `New`. Write the gams code in the editor window, and save within the project environment as a `.gms` file.
4. Click on the button with the icon of a page and an arrow. This would run the gams file, and create `.lst` and `.log` files.
5. Open the `.lst` file in the editor to see your results.

3 GAMS Input

In this section, we briefly discuss the structure and syntax of the GAMS input file. Only the most important features of the language are discussed to get you started. You must refer to the GAMS manual for detailed information.

3.1 Basic Structure:

To describe and solve a mathematical program using the GAMS language, the input file should

1. declare `SETS` of indices over which the problem is defined and assign its members,
2. declare the problem parameters and assign values in the form of `PARAMETER`, `TABLE` or `SCALAR`,
3. declare the `VARIABLES` and their types,
4. declare the `EQUATIONS` and define them,
5. declare the `MODEL`,
6. provide `SOLVE` statements,

7. provide instructions for `DISPLAY` of solutions.

The above sequence is a valid form for constructing a GAMS input file. The words in typewriter fonts are some of the special GAMS keywords. We describe the above steps in more detail using the file `example.gms` (see Appendix II).

3.2 Declaring Indices

The indices over which the mathematical program is defined are declared using the `SET` statement.

In the example (Appendix I), to declare the membership of index i to the set of stocks {1,2,3}, the GAMS statement is:

```
sets i/1*3/;
```

The asterisk `*` above means all integers between 1 and 3. Note the semicolon at the end of each description. Alternatively, we could have explicitly included the names of the stocks

```
sets i/IBM, WMT, SEHI/;
```

3.3 Declaring Parameters

To declare the data for problem parameters that are defined over one index (eg. s_i and d_j in the example), we use the `PARAMETER` statement. In `example.gms`, we have the following:

```
parameter rbar(i)
/
1        0.0260023
2        0.008100891
3        0.073774971
/;
```

The above statement declares the name the parameter `rbar` and its domain (index) `i`, and also assigns values corresponding to each index.

To declare parameters over double indices, the `TABLE` statement is used. For example to declare and assign values to the covariance parameter σ_{ij} in the example, we use the statement:

```
table sigma(i)
    1            2            3
 1  0.017087987  0.003298885  0.001224849
 2  0.003298885  0.005900944  0.004488271
 3  0.001224849  0.004488271  0.063000818 ;
```

Any blank entries in a table are interpreted as zeros.

Parameters that are not indexed can be defined using the `SCALAR` statement. For example to delare parameters B and R that has values of 1000.00 and 50.00, we would use:

```
scalar B /1000.00/
       R /50.00/;
```

Parameter values can also be directly assigned. However, the parameter and its domain has to declared first. For example, the parameter B discussed above could be entered as:

```
PARAMETER B;
B = 1000.00;
```

Similarly, the values for $\bar{r}_i$ could also be assigned as:

```
PARAMETER rbar(i);
rbar(''1'') = 0.0260023;
rbar(''2'') = 0.008100891;
rbar(''3'') = 0.0737749713;
```

Note that *you cannot declare the same parameter more than once.*

3.4 Declaring Variables

The decision variables of the problem and their types are declared using the `VARIABLE` statement. The type of a variable describes its allowed range, e.g. `POSITIVE` variables have the range 0 to $+\infty$ and `FREE` variables have the range $-\infty$ to $+\infty$.

In GAMS the objective function has to be defined in the form of a `FREE` variable that is not indexed. If the type is not declared the variable is assumed to be `FREE` by default.

In the example problem the problem variables x_i and the objective function variable are declared as:

```
positive variables x(i);
free variable z;
```

3.5 Declaring Equations

The constraints of the problem are declared using the `EQUATIONS` statement. This statement should declare the name and the index for each equation. Keep in mind that in GAMS the word `EQUATIONS` encompasses both equations and inequalities. For the example problem, the following statements declare the constraints of the problem:

```
equations obj, c1, b1;
```

Note that the objective function has to be expressed in the form of an equation with respect to the variable defined for the objective function.

Upon declarations, the constraints have to be defined. The equations are defined in the following order:

1. equation name as declared
2. the index set (if any)
3. the symbol `..`
4. the left-hand side
5. the relational operator (`=L=` for $\leq$, `=E=` for $=$, or `=G=` for $\geq$)
6. the right-hand side

Standard arithmetic operations such as exponentiation, multiplication, division, addition and subtraction are expressed through the symbols `**`, `*`, `/`, `+` and `-` respectively. These can appear on both sides of the relational operator. The summation operator ($\sum$) is expressed by the `SUM` statement which has two arguments, the index set and the summand.

The constraints for the example are defined as follows:

```
obj..  z =e= sum(i, sum(j, sigma(i,j)*x(i)*x(j)));
b1..  sum(i, x(i)) =l= B;
c1..  sum(i, rbar(i)*x(i) ) =g= R;
```

As another example, a constraint such as $\sum_i (x_{ij}^2 + \frac{1}{y_{ij}}) \geq C_{ij}$ for all j, can be defined as follows:

```
EQUATION CONSTRNT(j);
CONSTRNT(j)..  SUM(I, (X(I,J)**2 + 1/Y(I,J) )) =G= C(I,J);
```

3.6 Declaring the model

The `MODEL` statement declares which of the equations to be considered for a particular model. This declaration gives a name to the model along with a list of equations. The statement in the example:

```
model example /all/;
```

declares the model EXAMPLE consisting of ALL the declared equations. This statement could also be written as:

```
model example /obj, b1, c1/;
```

Note that indexes (if any) are omitted in the list of equation names.

3.7 Calling the solver

Once the model is complete, we can call a built-in solver to solve it. A valid format for the `SOLVE` statement is:

1. The keyword `SOLVE`
2. The model name
3. The keyword `MINIMIZING` or `MAXIMIZING`
4. The name of the variable to be optimized
5. The key word `USING`
6. Solution procedure, such as `LP` for Linear Programming, `NLP` for Nonlinear Programming and `MIP` for Mixed-Integer or Integer Programming.

3.8 Displaying Solutions

The output of a GAMS run is written on to a list file. The solution of all variables and the objective function value along with much more additional information is written to this file. Optionally, we can also request the solutions of specific variables by using the `DISPLAY` statement. These are also written into the list file. In the example, the statement:

```
display x.l;
```

produces a table of solutions for the optimal values of the variables in the list file. The `.L` extension means the final levels of these variables. If we were only interested in displaying the optimal amount to invest in IBM, we could write:

```
DISPLAY X.L(''1'');
```

3.9 Important Remarks

The following points should be remembered when writing a GAMS input file:

1. The ordering of the statements should be such that no entity (variable, parameter or equation) is used before it is declared.
2. Each statement should be terminated by a semi-colon.
3. The name of an entity should begin with a letter followed by up to nine more letters or digits.
4. Any line beginning with an asterisk (*) is considered as a comment.
5. The compiler is case-insensitive, so you are free to use either upper or lower case.

6. To declare the objective function, you must first create a free variable that is to be optimized. Then the objective function has to equated to this variable by means of an equation.

4 GAMS output

The output of a GAMS run is written onto a list file (with a `.lst` extension). This file has a lot of information. The pertinent information for our purposes would be the final levels of the problem variables, which also consists of the objective function value. These can be separated out from the rest of the list file by means of the `DISPLAY` statement in the input file. For our example, the list file `example.lst` contains following lines indicating the solution owing to the `DISPLAY` statement in the input file:

```
----     34 VARIABLE  x.L

1 497.669,    3 502.331
```

Note that errors in the input file are indicated on the list file. The error messages start with `***` with a `$` directly below the point where the compiler thinks the error occurred. The `$` is followed by a numerical error code, which is explained later in the list file.

5 Concluding Remarks

This article is supposed to get you started in using GAMS. I have tried to cover most of the features that will be required for our class. There are a large number of additional features in this software that have not been discussed. You should refer to the manual for any further information.

Appendix I

Mathematical Formulation:

Indices:
i = Stocks (IBM, WMT, SEHI)

Parameters:
B = Investment budget.
R = Desired expected return.
$\bar{r}_i$ = Expected return of stock i.
σ_{ij} = Covariance of the return of stock i to that of stock j

Variables:
x_i = Amount (\$) to invest in stock i.

Formulation:

$$\begin{aligned} \min \quad & \sum_{i=1}^{3}\sum_{j=1}^{3} x_i x_j \sigma_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{3} x_i \le B, \\ & \sum_{i=1}^{3} \bar{r}_i x_i \ge R, \\ & x_i \ge 0 \quad i = 1, 2, 3. \end{aligned}$$

Data:

$B = 1000.00, \quad R = 50.00.$

σ_{ij}	IBM	WMT	SEHI
IBM	0.017 087987	0.003298885	0.001224849
WMT	0.003298885	0.0059 00944	0.004488271
SEHI	0.001224849	0.004488271	0.06300 0818
$\bar{r}_i$	0.026	0.008	0.074

Appendix II

Here is the GAMS input file for the example problem in Appendix I. Note that GAMS is case-insensitive.

File `example.gms`:

```
sets i/1*3/;
alias (i,j);

parameter rbar(i)
/
1        0.0260023
2        0.008100891
3        0.073774971
/;

table sigma(i,i)
         1              2              3
1        0.017087987    0.003298885    0.001224849
2        0.003298885    0.005900944    0.004488271
3        0.001224849    0.004488271    0.063000818
;

scalar B /1000.00/
       R /50.00/;

positive variables x(i);
free variable z;

equations obj, c1, b1;

obj.. z =e= sum(i, sum(j, sigma(i,j)*x(i)*x(j)));
b1..  sum(i, x(i)) =l= B;
c1..  sum(i, rbar(i)*x(i) ) =g= R;

model example /all/;

solve example minimizing z using nlp;

display x.l;
```

Appendix III

Here is the GAMS output file for the example in Appendix I.

File `example.lst`:

```
GAMS Rev 120  Windows NT/95/98                  01/14/02 07:07:54  PAGE     1
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
C o m p i l a t i o n


   1  sets i/1*3/;
   2  alias (i,j);
   3
   4  parameter rbar(i)
   5  /
   6  1        0.0260023
   7  2        0.008100891
   8  3        0.073774971
   9  /;
  10
  11  table sigma(i,i)
  12           1              2              3
  13  1        0.017087987    0.003298885    0.001224849
  14  2        0.003298885    0.005900944    0.004488271
  15  3        0.001224849    0.004488271    0.063000818
  16  ;
  17
  18  scalar B /1000.00/
  19         R /50.00/;
  20
  21  positive variables x(i)
  22  free variable z
  23
  24  equations obj, c1, b1;
  25
  26  obj.. z =e= sum(i, sum(j, sigma(i,j)*x(i)*x(j)));
  27  b1..  sum(i, x(i)) =l= B;
  28  c1..  sum(i, rbar(i)*x(i) ) =g= R;
  29
  30  model example /all/;
  31
  32  solve example minimizing z using nlp;
  33
  34  display x.l;


COMPILATION TIME     =        0.000 SECONDS    0.7 Mb      WIN198-120
GAMS Rev 120  Windows NT/95/98                  01/14/02 07:07:54  PAGE     2
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
Equation Listing    SOLVE example USING NLP FROM LINE 32


---- obj  =E=
```

```
obj..  (0)*x(1) + (0)*x(2) + (0)*x(3) + z =E= 0 ; (LHS = 0)


---- c1  =G=


c1..  0.026*x(1) + 0.0081*x(2) + 0.0738*x(3) =G= 50 ; (LHS = 0, INFES = 50 ***)


---- b1  =L=


b1..  x(1) + x(2) + x(3) =L= 1000 ; (LHS = 0)
```



```
---- x


x(1)
                (.LO, .L, .UP = 0, 0, +INF)
       (0)      obj
        0.026   c1
        1       b1

x(2)
                (.LO, .L, .UP = 0, 0, +INF)
       (0)      obj
        0.0081  c1
        1       b1

x(3)
                (.LO, .L, .UP = 0, 0, +INF)
       (0)      obj
        0.0738  c1
        1       b1


---- z


z
                (.LO, .L, .UP = -INF, 0, +INF)
        1       obj
```



```
MODEL STATISTICS
```

```
BLOCKS OF EQUATIONS       3     SINGLE EQUATIONS        3
BLOCKS OF VARIABLES       2     SINGLE VARIABLES        4
NON ZERO ELEMENTS        10     NON LINEAR N-Z          3
DERIVATIVE POOL          11     CONSTANT POOL          14
CODE LENGTH             153


GENERATION TIME      =        0.130 SECONDS    1.9 Mb      WIN198-120


EXECUTION TIME       =        0.130 SECONDS    1.9 Mb      WIN198-120
GAMS Rev 120  Windows NT/95/98                  01/14/02 07:07:54  PAGE      5
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


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

     MODEL   example             OBJECTIVE  z
     TYPE    NLP                 DIRECTION  MINIMIZE
     SOLVER  CONOPT              FROM LINE  32

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE            20742.0766

 RESOURCE USAGE, LIMIT          0.170      1000.000
 ITERATION COUNT, LIMIT         4         10000
 EVALUATION ERRORS              0             0


    C O N O P T     Windows NT/95/98  version 2.043F-007-043
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default control program.


 ** Optimal solution. There are no superbasic variables.


 CONOPT time Total                       0.070 seconds
   of which: Function evaluations        0.000 =   0.0%
             Derivative evaluations      0.000 =   0.0%

 Work length =    0.05 Mbytes
    Estimate =    0.05 Mbytes
    Max used =    0.04 Mbytes

                      LOWER     LEVEL     UPPER    MARGINAL

---- EQU obj            .         .         .        1.000
---- EQU c1          50.000    50.000     +INF     968.646
```

```
---- EQU b1             -INF   1000.000  1000.000    -6.948


---- VAR x

     LOWER     LEVEL     UPPER    MARGINAL

1      .      497.669     +INF       .
2      .         .        +INF      6.894
3      .      502.331     +INF       .

                      LOWER     LEVEL     UPPER    MARGINAL

---- VAR z              -INF  20742.077     +INF       .
GAMS Rev 120  Windows NT/95/98                   01/14/02 07:07:54  PAGE      6
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




**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
                             0     ERRORS
GAMS Rev 120  Windows NT/95/98                   01/14/02 07:07:54  PAGE      7
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
E x e c u t i o n


----     34 VARIABLE  x.L

1 497.669,    3 502.331


EXECUTION TIME       =        0.040 SECONDS    1.4 Mb      WIN198-120


USER: Shabbir Ahmed                                  G010319:1121AE-WIN
      Georgia Tech, Industrial and Systems Engineering          DC3253

**** FILE SUMMARY

INPUT      C:\DOCUMENTS AND SETTINGS\SAHMED\MY DOCUMENTS\ISYE6673 SPRING 02\PORT
            EXAMPLE\EX.GMS
OUTPUT     C:\DOCUMENTS AND SETTINGS\SAHMED\MY DOCUMENTS\ISYE6673 SPRING 02\PORT
            EXAMPLE\EX.LST
```