

[GAMS - Undocumented features and usage tips](#)
[Index of topics](#)
[Index of Commands](#)
[Down Load Adobe Acrobat version of this document](#)
[Download Adobe Acrobat viewer](#)
[E-mail McCarl](#)

Use a Small Model

Debugging is often best accomplished using a small model version. For example, if one has a transportation problem with a 100 source points and 100 destination points, one can completely debug the model structure, report writing and calculations using two supply points and two demand points. One can do this via one or two mechanisms. First, one can develop a small data set which has all the structural elements of a full data set but much less data thus only supply for two supply points. Second, one can define a model to work over subsets of the full data set and greatly restrict those subsets. In this case, one might define all 100 supply points in a set called SUPPLPOINT, but write the model with a subset SUPPLYPT (SUPPLPOINT) then cause the model to only run over the sub set. Later after the model is debugged SUPPLYPT can be expanded to the full set of points.

Avoiding Excessive Execution Time and Memory Use

Models may take excessive time and/or memory in execution. There are several GAMS features which allow one to discover additional information about execution characteristics. These are separable into features which allow execution performance evaluation and features which yield information on memory use.

Root Causes Frequently GAMS users run into memory and/or execution difficulties because of run away dimensions. If the sets I, J, K, L, M, N, and O all have ten elements, any of the following statements would cause very slow execution and/or memory limit induced failure.

```
X(I, J, K, L, M, N, O) = 5;  
X.UP(I, J, K, L, M, N, O) = 10;  
X.SCALE(I, J, K, L, M, N, O) = 20;
```

In fact, any statements would cause such problems wherein a variable, parameter or constraint are unconditionally defined over these dimensions. If such multidimensional items are needed either bigger, faster computers or conditional (\$) statements limiting the cases considered must be used.

Time for Statement Execution and Cases Generated One can request that GAMS report information on statement execution characteristics. The resultant information tells how long it takes to execute a particular statement and the number of set elements over which the statement is executed. This is requested using the PROFILE command. The PROFILE command is invoked either through the use of the command line parameter.

```
GAMS FILENAME PROFILE = 1
```

or through the use of the Option command

OPTION PROFILE = 2;

The number after PROFILE tells how deep within if statements and LOOPS to provide timing information. When the PROFILE is set to 2 then the analysis includes the first level of statements within IF statements and loops. If one wants all statements, larger numbers can be used.

The PROFILE command causes output to be generated giving the time to execute each statement, the number of cases executed and the cumulative execution time up until that statement.

PROFILE output can be excessive since, by default, every statement is reported on. One can limit the reporting to only statements with execution times in excess of a user specified tolerance (say in excess of 1.5 seconds) by using the command

OPTION PROFILETOL = 1.5

Memory Use Dumps One can get information on memory use by item in the GAMS execution by inserting the command

OPTION DUMPSYM

which gives relative measures of memory use at the point of statement insertion. The command

OPTION MEMSTAT

gives a dump of total memory use at the point of statement insertion. The resultant output is difficult to interpret. Only the Total memory use row should generally be examined.

Model Optimality Status

GAMS can produce overwhelming amounts of output. Users may suppress the solution output as discussed under the output heading. However, one still may need access to the optimality status of the model. This may be done by displaying or using the modelname.modelstat parameter. Namely, if we had a parameter called X then we could set X as follows

X = Modelname.modelstat;

Display X;

and display X or we could directly display the model status item

Display Modelname.modelstat;

The meaning of the numerical values of modelstat are explained on pages 117-118 of the GAMS manual (only values 1 or 2 indicate optimality).

Save/Restart

An important feature in GAMS to manage output, allow use of advanced bases and potentially allow model distribution without revealing source code is SAVE/RESTART. Mechanically, these features allow the user to break a sequence of GAMS code into multiple pieces and allow later pieces to run as if they were attached to the earlier pieces. Thus, if one for example had a model up through a SOLVE statement in one file and a second file with report writing features utilizing optimal levels and marginals to compute output reports, one could use the command sequence

```
GAMS File1 S = Name  
GAMS File2 R = Name
```

where “Name” gives the name of an intermediate saving file. In this case File2 would run just as if it was appended to the back end of File1. However, none of the File1 output would appear in conjunction with File2 so File2.lst might only contain the output from a few display statements allowing one to create a much shorter output file. SAVE/RESTART also allows solves in subsequent files to start from the optimal basis in the predecessor file. This is commonly done in comparative statics analysis or when using GAMSCHK so one avoids the solution time required to manufacture the first solution. Finally, as discussed on the GAMS home page, one can use these features to distribute a “compiled” or “run time” version of a model without explicitly allowing access to the entire code. Furthermore, GAMS corporation is working on a feature called XSAVE which will produce platform independent restart files.

Problems with Presolves

Difficulties can arise with the OSL and CPLEX PRESOLVE procedures which simplify the problem by eliminating redundant equations or other predetermined structural elements. Problems may be simplified to the point that the subsequent solve fails or an infeasible solution is found. Both of these results cause the solver to return little or no information. Users may wish to obtain more information in these cases and may desire to suppress the presolve. This is accomplished by using solver option files as discussed above. In particular, when one wishes to suppress the presolve with OSL, what one does is create the OSL.OPT file with the line

```
PRESOLVE -1
```

In CPLEX one would create CPLEX.OPT with the contents

```
PRESOLVE 0
```

In both cases the presolve is suppressed and the solver deals with the full model. One may also use the option file to ask CPLEX to generate additional information about infeasibility causes by including the command

IIS 1

Report Writing

One feature lacking in description in the original GAMS reference manual involves report writing. Here material is given on using solution information in report calculations, customizing reports, altering item ordering and overcoming merge and replace difficulties.

Using optimal solution information in reports One can, for example, set up a parameter and in turn, after a solve calculate items involving optimal solution parameters. For example, if one had either a variable and equation named X then the following four statements could be used

```
z(j,"optlevel")=x.l(j);
z(j,"marg")=x.m(j);
display x.l;
res(i,j)=a(i,j)*x.l(j);
```

The first statement puts the optimal levels of X then into the Z array, the second picks up the reduced cost or shadow prices. The third statement displays the nonzero level values of X. The fourth computes, for example, a table of resource usages by resource(i) and variable (j) assuming there is an equation in the model that is of the standard form of the resource allocation $AX \leq b$.

Reformatting Output GAMS output can be customized using PUT files or display option modifiers. The ultimate degree of customization is gained by using PUT files as discussed in the GAMS 2.25 users guide on pages 275-277 or thru the PUT FILE document that is distributed through the GAMS web page. PUT files allow highly customized output but require more advanced programming skills. Individual display statements may be customized using the option syntax discussed in the manual on pages 146-147.

Item ordering Report item ordering can be frustrating. GAMS reports set items in the order in which they are initially defined. Furthermore, if an item name in a set has been seen somewhere else then it will be arrayed before items whose names appeared for the first time at a later stage. So if one defines the following

```
SET ONE /A1, A2, TOTAL/
     TWO /B1, B2, TOTAL/
```

then whenever displaying anything containing set TWO the entry labeled TOTAL will appear before B1 and B2 because the word TOTAL has been seen before. Users frustrated with this can do one of two things. First, they can come up with a unique name for TOTAL in later sets (i.e.

TOTALS) which if unique will insure that item will always be at the bottom. Second, you can add one dimension to the parameter and force sequencing . For example, if you add

```
SET NUMBER /R1*R100/  
PARAMETER ITEMTO DISP(NUMBER,TWO)
```

then associate all entries in ITEMTO DISP with R1 and the TOTAL element with R100, then any display statement involving ITEMTO DISP should have TOTAL coming out last.

Merge and replace difficulties GAMS has features which allows it to save information from a sequence of model solves. However when the same model is being solved in a comparative statics framework then these features can cause difficulties. Suppose we are solving a model with the variable X in it and X is present in the first solve but is eliminated via \$ controls in the second. In that case GAMS would retain all solution information for X between solves and any report writing using X.L after the second solve would be using the optimal X values from the first solve. This can be avoided using the OPTION SOLVEOPT=REPLACE; (as discussed on page 198 of the GAMS manual) as opposed to the default merge option with variables that are eliminated in a run will be obtained. However, one should note that the replace OPTION does not absolutely guarantee proper function. Mainly, if a particular variable within a variable is deleted in a run its value will be retained. Thus, if one had a model with X1 and X2 in it where the 1 and 2 are two different set elements for the X set. Then if one solved the model originally with X1 and X2 both then thru \$ conditions removed X1 and solve the model again and displayed X.L one would find the X1 value from the solution was still present. This is a nasty feature which GAMS is attempting to rectify but now requires manual zeroing of such variables.

Advanced Basis

The solvers used in GAMS allow one to use an advanced basis. GAMS exploits this but not generally on the first solve. GAMS generation of an advanced basis relies on a stored incumbent solution which GAMS ordinarily will only have access to after a model has been solved at least once. Thus GAMS in general can only use an advanced basis in solves after a first model solution. Users wishing to suggest a basis for the first solve must provide an incumbent solution. This can be done via one of two mechanisms, manually or by use of GAMS BAS as distributed through this web page. When manually suggesting a basis users must provide level values for variables which should be basic and marginals for nonbasic variables as well as marginals on binding equations. The simplest way for most users to use an advanced basis would be to solve a model from a cold start then retain the basis using GAMS BAS or the SAVE/RESTART feature.

Unexpected Failures

GAMS can fail for a number of reasons. Often failure causes are indicated in messages in the LST file marked with ****. Examples of such messages and their repair are listed below

Message Text	Repair Mechanism
Solver Status Resource Interrupt or Resource Limit Exceeded	OPTION RESLIM=100000;(or a larger number)
Solver Status Iteration Interrupt or Iteration Limit Exceeded	OPTION ITERLIM=100000;(or a larger number)
GAMS terminated due to limits in PRESCAN009 or Maximum Executable Code Size Exceeded	Add CODEX=1 to GAMS command line or Reduce amount of code included in Loops

One may also run out of solver specific items. For example, exceeding the size of the work space for the Hessian matrix in MINOS or having difficulties caused by the presolves as discussed above. In such cases one should consult the solver document for a listing of solver specific parameters. These documents can be found on the GAMS web page or, for MINOS and ZOOM, in appendix D of the GAMS manual.

Difficult errors can also occur with excessive iterations, timing or memory use. Better problem scaling or degeneracy resolution may be needed.

Restricting the Amount of Output

GAMS output can be excessive particularly for new users. One can reduce the volume of using an appropriate mixture of the following commands:

Command	Function
\$OFFLISTING/\$ONLISTING	Suppresses the echoing of all input commands after the issuance in the input string of the OFFLISTING until the code segment ends or an ONLISTING is found
OPTION LIMROW=0;	Suppresses the listing of the equations in the programming model
OPTION LIMCOL=0;	Suppresses the listing of the variables in the programming model
\$OFFSYMLIST OFFSYMREF	Suppresses the symbol list and cross-reference map.
OPTION SOLPRINT = OFF;	Suppresses the print of the solution. Used when one is doing further report writing and does not wish to output the detailed solution.

One can also manage the output by for example putting display statements within IF conditions only causing output to occur if mandated by the results or by using the SAVE RESTART structure with a few display statements in the final file leaving the rest of the output in

a larger ordinarily unexamined file.

Putting Slack Variable Information in the Output

One can cause the equation output to contain slack variable values rather than levels by inserting the option command

```
OPTION SOLSLACK = 1;
```

Gaining Ranging Information

Post optimal optimality analyses in terms of ranging can be obtained by using the POSTOPT features explained on the GAMS home web page.

Bad Model Results

When solving models one can come up with models which

- terminate at an intermediate optimal
- exhibit excess iterations without significant progress;
- are reported as unbounded
- are reported as infeasible
- have an optimal but unrealistic solution.

Each event mandates different actions

Stuck or Excessively Iterating Models Model can iterate excessively without making significant progress becoming “struck.” In such cases one should investigate the scaling characteristics of the problem and also consider adding small random numbers of magnitude say a 1/1000 of the smallest expected variable value to right hand sides on potentially reluctant equations. Note that this is done automatically by CPLEX and OSL when they “perturb the solution”.

Unbounded In the case of an unbounded model the general best strategy to use is to place a large upper bound or very small lower bound on the variable being maximized (minimized) and then solve the problem. If that variable comes out equaling its bound upper bound then one should look through and try to find out where that objective function value is coming from. This can be done using the GAMSCHK mode looking at the objective function equation only to see where the large value is coming from.

One can also look at the output and find the variables marked NONOPT. However, one should note that when the variables marked POSTOPT at some point generally includes not only the variable that is NONOPTIMAL but also all the variables that have not been resolved in the optimality test at that point so one of the variables is the variable is the one that is the

NONOPTIMAL features in GAMSCHK under the NONOPT procedure help identify these.

Infeasible Yet another condition which can occur is an infeasible model solution. In this case, you should add artificial variables to any rows which are not feasible when all the variables are set to zero. This includes the quality rows with non-zero right-hand side \geq with negative right-hand sides. \leq with positive right-hand sides. GAMSCHK nonopt will help identify these and in turn when the model with the artificials is run GAMSCHK NONOPT will help identify the equations that are involved with the infeasibility.

Optimal but unrealistic This is the hardest case to deal with and implies that there is some sort of error in the model structure. GAMSCHK has been written to help people debug such cases but nothing can be written in general in these notes about how to do that.

Output Control - Making the Output Fancier

There are three ways to make GAMS output fancier. The first involves using the formatting capabilities of the DISPLAY statements as discussed on pages 146-147 of the BASE GAMS manual. The second involves PUT files as discussed in the gray appendix pages or in the more extensive PUT FILE released is on the GAMS home web page. The third involves using the spreadsheet interface as discussed on the GAMS web home pages.

Inconveniences

There are a few inconvenient features in GAMS. The first involves paging and the PUT files. When one is using PUT FILES the paging does not automatically work with PC based printers. This is correctable by specifying the option PC=3 as discussed on page 286 of the GAMS manual.

Second, when debugging a model which is running out of time or memory use operating systems usually cause the LST files to end well before point of program fault. Thus, one has to hunt to find the right place. What one has to do is selectively eliminate execution lines (using the ONTEXT/OFFTEXT features) starting at the end the file until one finds a part of a file that will work. Then one should slowly add back in statements until the statement that isn't working is found. This also may involve running a small model as opposed to large models to allow model debugging.

Finding Out More About the File Structure

When GAMS models are quite large input parameters can be defined in multiple places and files. In such a case, it can be difficult to find all definitions and uses. One can utilize the optimal file command RF as discussed in the GAMS web page about the command line options. One can also, use GAMSMAP which is distributed on the associate web as an aid.

External File Inclusion

Users often may wish to break their input into several files some which contain particular kinds of data, others of which contain model code etc. This can be done by four mechanisms. First, one can use the restart and save features discussed above or in the GAMS manual to break the model into pieces. Thus, the data piece might be separated from the model piece so clerical personnel can work on the data piece to do revisions without altering anything in the model. Second, one can use INCLUDE files as discussed in the gray pages of the 2.25 document on page 273 where one simply includes a set of GAMS instructions into the code. Third, one can use the BATINCLUDE command to include a set of files with replaceable parameters as briefly discussed in the GAMS manual on page 273. Finally, one can use the spreadsheet link as discussed on the GAMS web page.

Of these only the BATINCLUDE merits are further discussion. Suppose one wished to write a simple include file which displayed a multi-dimensional array while controlling formatting. One could do this by creating a file which for simplicity call IT which looks as follows:

```
OPTION %1:2%:3:%4; DISPLAY %1;
```

where %1 is the first parameter passed through the BATINCLUDE, %2 the second, %3 the third, and %4 the fourth. This code displays the data named in the first parameter (%1) with the decimal places specified in the second parameter, the number of items in each row specified in the third parameter and the number of column items as in the fourth. Example uses include the following two statements.

```
$BATINCLUDE IT DATAONE0 2 3  
$BATINCLUDE IT DATATWO2 1 1
```

These statements would result in the display of array DATAONE with zero decimal places and two items in each row and three in each column definition. Then array DATATWO would be displayed but with two decimal places and with one item in each row and column.

Problems with PC's

Using GAMS on a personal computer leads to a number of problems because of non-standardized software operating systems, ROMS, peripherals, etc. The biggest problem is often conflicts between memory managers and the way that GAMS is set up. GAMS Development Corporation has a page under their documentation section on how to avoid this. Generally, what can be said is that memory managers like EMM386, save memory resident utilities, and virtual disk software can cause conflict with GAMS. One generally needs to clear out to a very simple

Autoexec.Bat.

and

Config.sys

Also, GAMS has path name restrictions. Only short path names (40 characters or so) can be used. If one is having difficulty one needs to make sure that GAMS execution directory is not too deep in the path. Thus, either shorten up the path names to some element one to about 40 characters to the whole length.

Scaling

Users may find themselves in a position where the solver is having great difficulty due to the numeric characteristics of the problem. In such a case it is usually desirable to do manual scaling. GAMS assists manual scaling using two features.

First to invoke scaling one must insert the command

```
modelname.scaleopt=1;
```

where the MODELNAME is that identified in the model and solve statements .

Second for the variables and equations to be scaled one must insert statements of the form

```
varname.scale(setelements)=k;
```

```
eqnname.scale(setelements)=k;
```

where varname is the name of one of the problem variables

eqnname is the name of one of the problem equations

setelements are the associated set elements

k is any number or calculated parameter name

This scaling results in all the coefficients associated with a variable or equation being divided by the scaling factor. GAMS automatically descales all solution information so the scaling does not affect the solution output. However the LIMROW/LIMCOL output from GAMS does display the elements after scaling. An option in GAMSCHK controls whether the data displayed are before or after scaling.

In a model named farm with

variables named plant and harvest

equations obj and resource

sets crop , plantdate, and item

a defined parameter called scalval

the following scaling sequence could be used

```
farm.scaleopt=1;
```

```
plant.scale(crop,plantdate)=1000;
```

```
plant.scale("corn",plantdate)=400;
```

```
harvest.scale(crop)=scaleval(crop);
```

```
obj.scale=1000000;  
resource.scale(item)=777777;
```

Matching Set Elements

New features in GAMS allow one to introduce conditional statements controlling execution in cases where certain items match up . The syntax involves using the commands

```
sameas(setelement1,setelement2)  
or  
diag(setelement,setelement)
```

the sameas command returns a true false indicator which is true if the text string defining the name of set element 1 equals that for setelement 2 and false otherwise. DIAG returns a 1 under equality and a zero otherwise

For example

```
x=sum((i,j)$ (not same as(i,j)),z(i)*z(j));  
or  
x=sum((i,j)$ (diag(i,j) eq 0),z(i)*z(j));
```

would exclude the cases where $i=j$ from the sum

while

```
x=sum((i,j)$ (same as(i,"case1") or same as(j,"case2")),z(i)+z(j));
```

would only include cases where the text for i equaled the string "case1" or the text for j corresponded to "case2"

Option files

Solver function is controlled by Option files. For example, option files can be used to control:

- nonlinear scaling in MINOS5
- the incidence of the presolve in CPLEX and OSL
- integer branch and bound strategies in OSL and CPLEX
- the use of an infeasibility finder in CPLEX
- choices of optimizer in GAMSCHK/GAMSBAS

Users should get the solver manuals to obtain a full display of potential option file contents.

The use of the option file is controlled by the command
model.optfile=n;
when n =1 the option file used is solvername.opt

when $n > 1$ and $n < 10$ the name is solvername.opn i.e. op1,op2 etc.
when $n > 9$ and $n < 100$ the name is solvername.on i.e. o10,o99 etc.

Thus, when solving a model named FARM with the solver CPLEX if one enters the command

```
farm.optfile=1
```

then one would expect to find the file cplex.opt containing contents such as

```
presolve 0
```

or if using MINOS5 then the file would be named minos5.opt and could contain

```
scale nonlinear variables
```

while

```
farm.optfile=2;
```

would suggest the file osl.op2 would be present with contents like

```
presolve -1
```