Bruce McCarl's GAMS Newsletter Number 40

May 2017

It has been nearly a year since I made a Newsletter. Things have been busy. Anyhow this Newsletter addresses the following

Contents

1 New Features

A number of new features have arisen involving commands within gams, the documentation, and the solvers. Here I review what I consider to be the more important ones. Note also that the expanded user guide is being revised its same time to include these features.

1.1 New features within GAMS

The release notes since the last newsletter show a number of new but relatively minor commands or features. In my mind the more useful of them follows

- New commands **Break** and **Continue** let you jump out of the execution sequence within a **LOOP**, **WHILE**, **REPEAT** or **FOR** group of statements.

    - **Break** causes the repeated execution to be terminated and can have an argument where break **1** causes the innermost sequence to terminate, break 2 the next most inner and so forth.
    - **Continue** causes execution to Jump to the end of the inner most control structure skipping all commands between the continue and the end of the loop. An example (breakcontinue.gms) was placed in the Expanded GAMS user guide although the example revealed a bug in Break that will hopefully be repaired in the next release.
- A new command line option **fileStem** allows you to use a different stem name for LST, LXI, LOG and default GDX files. Thus if you run myfile.gms but use the command line parameter filestem=anothername then the LST file etc will be named anothername.lst. This allows users to save these files with a unique name when one is running multiple cases of single GAMS program in the same directory.
- **MCPRHoldfx** has been introduced that if set to one results in a printout of the rows in an MCP problem that are complementary with variables that are held fixed by a .fx command and when the holdfixed command is employed are not included in the model passed to the solver. In turn in this case a number of equations are also dropped to maintain a square system and this command results in a list of them. In turn that list helps advanced users sort out difficulties with matching up complementarities between variables and equations. This can be set using an option statement, a command line parameter, or model attribute. Its default is zero resulting in no print out of the list and when set to one such a list is created.
- A **multi threading** option is available that causes GAMS to use multiple processor threads while also employing efficient in-memory communication with a solver. It is imposed by setting the option or model attribute **SolveLink** to 6. Other than that, the multi-threading facility works in the same way as the Grid facility. Additional documentation is available on the GAMS web page here.
- The option ThreadsAsync controls the number of threads to be used when SolveLink=6. Setting this to zero uses all available threads. Setting it to a positive number gives an upper limit on the number of threads to use and setting it negative tells how many threads to reserve. The default is -1 reserving one thread. Again additional documentation is available on the GAMS web page here.
- **AsyncSolLst** is an option, or a command line parameter that when set to 1 causes solver output from grid or multi-threaded runs to go to the LST file. The default value is 0.
- The **SolveLink** option and model attribute can now equal 7 which causes in core passage to the solver without use of temporary files with GAMS waiting for the solver to return the solution. This is included mainly for debugging purposes as is solvelink=4.
- The solution report summary now contains a count of the number of variables and equations where the level was projected to one of the bounds since it was within the tolerance specified by tolProj provided that number is greater than 0.
- A function **numCores** returns the number of logical cores in the system. It can be used at compile time using the syntax $eval myCores numCores which places the number of cores into the control variable mycores. Inside GAMS one can set a scalar to the number of cores using myscalar=numcores;

- A dollar control splitOption splits a parameter sequence in the form "`-a3=0`" or "`-a2=3.14`" or "`/opt=val`" or "`/opt val`" (you can interchangably use – or /) into two parts - the parameter to be changed and its value. This is designed for use in BATINCLUDE statements where one wants to alter parameter when invoking other executables. Details are given here.

1.2 New developments in Documentation

The documentation distributed with GAMS systems that is designed to work offline now provides search and keyword indexing functionality.

The table of contents for browsing the GAMS model libraries is now available offline in HTML format.

1.2.1 GAMS Support Wiki

GAMS has a Wiki that contains answers to common questions that arise during support calls. It is located at http://support.gams.com/start. This wiki was originally designed for usage by gams staff during support calls but has been made available on the Internet. Its content is not necessarily stable but may answer number of users questions.

1.2.2 Videos

GAMS has a number of videos on various topics that are present on YouTube at https://www.youtube.com/user/GAMSLessons .

The titles of the videos are

GAMS and Matlab GDXMRW tools RGDX and WGDX
GAMS and Matlab Setup and Introduction to GDXMRW
How to Install the Native GAMS Version on Linux
GAMS and Excel - Using GDX to Transfer Data
Using a Solver Option File
An Introduction to Sets in GAMS
Install the Windows Version of GAMS on a Mac by Using Wine
A Brief Introduction to Modeling in GAMS
GAMS License File Installation and Component Review
How to Install GAMS on Windows

1.2.3 An emerging new user documentation

GAMS has been working for the last couple of years on improving its documentation with the main available product to date being the changes on the webpage and the inclusion of a lot more documents on individual features that are accessible through the webpage and through the associated search.

This involves creation of a totally new and comprehensive user guide that is also being structured so it's as terse as possible. This is likely to fully materialize in the next year or so plus I think that parts of it have been emerging in the on and off line documentation that can be attached through the IDE. The GAMS people see this new documentation as a replacement for the Expanded GAMS user manual but I am not so sure.

I believe the two documents will be complementary with the guides each serving a mixture of different audiences or different needs. I believe the Expanded Guide will be used more for those who want to see a more extensive "expanded" discussion with examples and cross references as they try out new features plus those who are willing to read. On the other hand, those who want to get straight to the point say with recalling the exact syntax for unfamiliar or seldomly used command may find themselves making greater use of the terse one. But I believe with today's document delivery and search capabilities that there is not so much value in terseness or loss in a more extensive presentation. I use the McCarl guide frequently to update me on small little details in syntax.

I also believe the new terse guide will have one another characteristic which is GAMS staff will be updating it. Therefore, it may have coverage things that I don't know about which never make the release notes like the macro and & discussion above. On the other hand, my experience has been that the GAMS staff explanations are frequently so terse that no one knows what they mean except the person who wrote them and us need substantial elaboration. Almost every time I update the expanded user guide I have significant questions about features mentioned in the release notes and end up in a discussion with one or more GAMS staff until I can finally understand as must as possible on: a) what the feature really does, b) exactly how to specify it in GAMS code and c) the circumstances under which it is desirable to use it.

Time will reveal the fate of the Expanded Guides but for now I will keep it up to date. I should also admit that one I've been keeping it up for more than 15 years that I really doubt I'll be doing it in another 10. Thus If there is anyone who would like to help and is also highly adept at GAMS they should contact me.

1.3 New developments in Solvers

Many of the solvers have upgraded features incorporated in newer executables. Most of the items mentioned in the release notes just say a new library was incorporated into GAMS. Among the more substantial additions mentioned are

- CPLEXD now contains an implementation of the Benders Decomposition Algorithm. It also has an alternative program called the conflict finder that finds the cause of infeasibilities in models. The conflict finder replaces the IIS procedure. The conflict finder has new capabilities in that it works with discrete variables.
- A newer CONOPT version has been released and is named CONOPT4. According to the CONOPT4 solver manual this does not replace CONOPT3 but rather augments it. Specifically, CONOPT4 is designed to perform better on problems where CONOPT3 does not do so well. The solver guide enumerates a number of cases where it's use may be beneficial. Specifically it is recommended that users try it on models that
    - Take more than a few minutes to solve with CONOPT3;
    - Are larger CNS models(those with more than 100,000 variables and constraints);
    - Cause CONOPT3 two run out of memory.
    - Are ones where CONOPT3 ends it solution process with a locally infeasible solution
    - are ones where where CONOPT3 finds a large number of definitional constraints.
- DICOPT has an added Feasibility Pump primal heuristic (for convex MINLPs) that is used to start up and integer solution.

- GUROBI has a number of new solver control parameters. It also supports general constraints, indicator constraints and multi objective hierarchical optimization.
- Knitro has new speed and robustness features that arise when using BFGS or L-BFGS with the default interior-point method. It also contains a new algorithm for nonlinear mixed-integer models that is intended for small, potentially non-convex models with expensive function evaluations or integer variables that are not relaxable
- Lindo/LindoGlobal has speed and MIP Solver Improvements plus altered MIP preprocessing, Stochastic Solver Improvements, Improved cut management for Benders Decomposition, Better handling of multistage stochastic models which do not have full-recourse and Global Solver Improvements regarding bound tightening and linearization.
- LocalSolver has new features for solving near-linear (discrete or continuous) problems
- Mosek has improved presolve performance and a reprogrammed implementation of the pre-solve eliminator along with improved presolve performance on conic quadratic problems.
- SCIP changed the default LP solver to Cplex and has transitioned into a commercial solver that commercial clients can acquire but is free to academics.

1.4 Gams tools

GDXDUMP has a new command line option CSVAllFields to dump all fields (level, marginal, lower, upper, and scale) when writing a variable or equation symbol in CSV format.

2 Macros, a problem with line splitting and the magical &

I recently had problems with a macro and discovered both an issue and a previously hidden GAMS feature that avoided the issue. Let me illustrate this with a simple example. Suppose I formed a macro for the right hand blue colored part of the statement below

```
Y= (+12+3*x-0.005*x**2)$(x gt 0 and x <= 200)
```

Where when I formed it the macro was spread over several lines using the continuation symbol "\" as follows

```
$macro evalx(x)                          \
            (+12+3*x-0.005*x**2)         \
               $(x gt 0 and              \
                 x <= 200)
```

I then ran this and got an error. Examining the LST file I found the expanded macro was

```
y=(+12+3*x-0.005*x**2)$(x gt 0 andx <= 200);
```

where I found out that GAMS automatically ***stripped all beginning and ending spaces*** from the lines. This resulted in the macro expansion running the "and" together with the "x" so it was entered the compiler as "andx" which the GAMS compiler did not recognize and identified as an error.

I then tried several approaches to fix this and had concluded I could never end a macro line with some command which had to have a space after it like and, or, ne, Eq etc. I then asked the GAMS staff if there was a way to avoid the problem. Subsequently I was informed by Alex that this could be fixed by adding an **&** at the end of the line after at least one space as follows

```
$macro evalx(x)                          \
```

```
(+12+3*x-0.005*x**2)      \
   $(x gt 0 and &         \
     x <= 200)
```

where the **& *causes the space to be retained at the lines end***.

Also in the process I found out that using && caused GAMS during the macro expansion to remove any quotes that were present when in argument was placed in quotes in the statement referencing the macro. Thus if one had the macro above (renamed to evalx1) and called it with

```
Y=evalx1("x");
```

That without any fancy provisions this would cause an error because gams would choke on the quotes. However one can avoid that error by modify the macro as follows

```
$macro evalx1(x)                \
                    (+12+3* &&x-0.005*&&x**2)          \
                    $(&&x gt 0 and &&x <= 200)
```

note in this case one has to put in && just before every reference to any argument that users could've placed in quotes.

I also discovered in forming this example that one could not end a line with an & and then have next line start with && as I would need to do to preserve the spacing after the "and" command as illustrated in the first example. Thus, I reformed the macro as above where a following space was not critical.

3 Misbehaving Model – Infeasible

I was talking to the GAMS staff and they informed me as to what their most common support calls involve. One of them involves fixing bad results. In this and the next couple of newsletters I will cover how to diagnose problems within models that don't work right. Also in the next section I will cover an integer programming issue that commonly comes up

Years ago I started writing a book length item called - So Your GAMS Model Didn't Work Right: A Guide to Model Repair but life got busy and I never finished it. (If someone wanted to help finish it please contact me). Here I will adapt material in it on infeasible models for presentation here. So here we go

Infeasibility is always a possible outcome when solving models. Simplex based linear programming solvers handle infeasibility through a two or three step solution approach. First, there may be some presolution calculations which may determine a model cannot be made feasible (as done for example by the CPLEX and CONOPT PRESOLVEs). Second, there is usually a Phase I operation wherein the sum of a set of implicitly added artificial variables is minimized. During this phase, the problem is artificially rendered feasible. Third, if the artificial variable values are all driven to zero, then the problem is declared feasible and the solver turns to the real objective function and proceeds toward optimally.

However, the problem may be declared infeasible by the presolve or the Phase I. In such cases, the information content of the output differs between solvers and may not be very helpful.

3.1 Causes of Infeasible Models

Causes of infeasibility are not always easily identified. Solvers may report a particular equation as infeasible in cases where an entirely different equation is the cause. Consider the following example,

Max	50 x1	+50 x2		
	x1	+ x2	$\leq$	50
	50 x1	+ x2	$\leq$	65
	x1		$\geq$	20
	x1,	x2	$\geq$	0

In this example, the interaction between the constraint $x1 \geq 20$, the constraint immediately above it, and the nonnegativity condition on x2 render the model infeasible while the first constraint has nothing to do with the infeasibility. There may be several potential explanations as to why the infeasibility is present. The 65 on the right hand side of the second constraint may be a data entry error, perhaps a number in excess of 1000 was intended. Similarly, the 50 coefficient multiplying x1 in the second constraint may be an error with a number more like 0.50 or a negative entry intended. Third, the limit requiring $x1 \geq 20$ may be misspecified with the RHS really intended to be 0.20. Fourth, perhaps the X_2 variable should have been allowed to be negative. Fifth, there could be multiple errors involving several of the above cases. Runs with CPLEX, BDMLP and MINOS5 resulted in the marking of either the $x1 \geq 20$ or the second constraint as the infeasible item. This may or may not be a proper identification of the problem causing mistake.

This illustrates a general point that infeasibilities occur because of the interaction of multiple balance on variables and the restrictions imposed by equations. In more complex models a larger set of constraints and bounds could be involved and there may be thousands of other variable bounds and constraints that have nothing to do with the infeasibility. Thus, we need procedures to identify the infeasibility causing set of variable and equation restrictions. In turn then we can look for the root cause of the infeasibility in that narrowed down set.

3.1.1 Finding Causes of Infeasibility -- Basic Theory

There are two approaches I recommend for finding the set of infeasibility causing equations. The first approach relies on "artificial" variables and the second involves use of infeasibility finders that appear in a few solvers (i.e. the CPLEX conflict refiner, the IIS as in BARON, GUROBI and XPRESS; and the feasibility relaxation in CPLEX and GUROBI). I will only cover artificial variables here as their usage works with all solvers.

Artificial variables are covered in virtually every introductory linear programming course or book. An artificial is an added variable that did not appear in the original model which is structured to make sure that the where it is inserted can all always be satisfied. Additionally, the model objective function is modified to provide a strong incentive to drive the artificial variables to zero. There are two ways such incentives are entered: through the "Big M" penalty method or through the "Phase I/ Phase II" optimization approach. Here we only cover the Big M method. In that case we augment the above model with an artificial variable as follows

Max	50 x1	+50 x2	-1000000 A				
	x1	+ x2				$\leq$	50
	50 x1	+ x2				$\leq$	65
	x1		+	A		$\geq$	20

x1, x2, A ≥ 0

with A being the artificial variable and the -1000000 being the objective function penalty. Here we only add one such variable but more generally artificials would be entered for each model equation which is not be satisfied when all decision variables equal zero (i.e., when x1 = 0 the x1 ≥ 20 constraint is not satisfied). In general, the added artificials would have a large, undesirable objective function coefficient (the so called "Big M value") and an entry in an associated potentially infeasible equation.

3.1.2 Identifying Infeasible Causes

Now let us go introduce the procedure for identifying the set of infeasibility causing constraints and variable bounds. This is done by solving the problem with artificials added and then using the solution information to identify the infeasibility causing equation and variable bound set. Now we illustrate the procedure using the example.

A GAMS formulation of the above problem after including the artificial is

```
variable           objmax;
positive variables x1, x2, A
equations          obj,r1,r2,r3;
obj..   objmax =e= 50*x1 +50*x2-1000000*A        ;
r1..                  x1 +   x2           =L=  50;
r2..               50*x1 +   x2           =L=  65;
r3..                  x1              +A =G=  20;
model infe /all/;
solve infe using lp maximizing objmax;
```

Note here it is possible that a solver with a presolve can eliminate some equations and in turn possibly not report the proper shadow prices. (when using a solver with a presolve one might suppress it, for example using the CPLEX option preind 0 or the XPRESS option presolve 0).

The resultant relevant part of the Big M solution is

```
**** SOLVER STATUS     1 Normal Completion
**** MODEL STATUS      1 Optimal
**** OBJECTIVE VALUE        -18699935.0000

                       LOWER     LEVEL     UPPER    MARGINAL
---- EQU obj             .         .         .        1.000
---- EQU r1            -INF       1.300    50.000       .
---- EQU r2            -INF      65.000    65.000  20001.000
---- EQU r3           20.000     20.000     +INF   -1.000E+6

                       LOWER     LEVEL     UPPER    MARGINAL
---- VAR objmax        -INF  -1.870E+7      +INF       .
---- VAR x1              .        1.300     +INF       .
---- VAR x2              .         .        +INF   -1.995E+4
---- VAR A               .       18.700     +INF       .
```

Where note the marginals on r2, r3 and x2 are large.

The question then is: So what? When a linear program is solved, the optimum solution contains items which are influenced by the objective function parameters for the non-zero variables. In particular the GAMS marginals or more classically, the shadow prices, reduced costs and objective function value

So in this case with the artificial is in the basis, then relaxation of some of the right hand sides will cause that artificial to become smaller and they will have artificially large shadow prices. Similarly the reduced costs on some variables will be influenced by the presence of the artificial indicating that the artificial would get smaller if that variable could go below its lower bound (going negative in this case) or above its upper. Note this is the case for the marginals on r2, r3 and x2 which jointly indicates equations r1, and r2 plus $x2 \geq 0$ is the infeasibility causing set.

3.1.3 General procedure for finding the infeasibility causing set

The following gives the steps for finding infeasibility causes using the Big M, artificial variable approach.

Step 1 Identify the relevant equations and/or variable bounds for which artificials are needed to be added (details about this in next section)

Step 2 Add artificial variables to those equations and bounds. These artificials each have a Big M penalty in the objective function and an entry in a single constraint.

Step 3 Solve the model

Step 4 Examine the model solution. Where the marginals (the reduced costs for the variables and the shadow prices for the equations) are distorted by the presence of the artificials, identify those as the variables and equations to be examined for the cause of infeasibility (note once identified then the modeler needs to examine those in the context of the problem hopefully finding the issue).

Step 5 Fix the model and repeat the process if needed

There are several questions inherent in the above procedure. In particular: Where should artificial variables be added? How should the artificial variables be structured? and How does one find a "distorted" marginal? Each is discussed below.

3.1.3.1 Where Should one add Artificial Variables?

The places where artificial variables should be added can be determined in several ways. One could look at the model solution and enter artificials in the equations and/or variable bounds marked by the solver as infeasible. However, while this sometimes points to proper places, it does not always do such. The approach advocated here is to add artificials in all possible infeasible locations although a different approach is in order for newly modified models as discussed below.

Programming models will only be infeasible when setting all the decision variables equal to zero is not feasible. This occurs when: a) the interval between variable upper and lower bounds does not include zero; or b) equations appear which are not satisfied when all variables are set to zero. The equation cases which are not satisfied when the variables are set equal to zero are:

1) Less than or equal to constraints with a negative right side i.e. $x \leq -1$
2) Greater than or equal to constraints with a positive right side. i.e. $x \geq 1$
3) Equality constraints with a nonzero right side i.e. $x = 1$ or $x = -1$

Additionally, when the interval between a variable's lower and upper bounds does not include zero then those bounds need to be converted to constraints with artificials. This will occur when:

1) The lower bound is positive, or
2) The upper bound is negative

The ADVISORY and NONOPT -- IDENTIFY procedures in GAMSCHK have been written to create a list of all occurrences of these five cases.

3.1.3.2 Adding artificials in newly modified models

When a model that was feasible before has been newly modified and goes infeasible this raises a different artificial adding procedure. Namely, one can just add the artificials into the newly added constraints and/or bounds. Therein one may need to add artificials to newly added constraints or bounds that in fact are satisfied when all the variables are set to zero. This arises because the new constraints are possible members of the infeasibility causing set through their interaction with other constraints that previously could be satisfied. Here artificials would be added to the new =L= constraints with positive right hand sides or new =G= constraints with a negative right hand side. The exact structure these artificials follows the rules given in the next section.

3.1.3.3 Entering Artificial Variables in GAMS

Once one has found where to add the artificial variables one still has to address: How they should be added? and What should they look like? I recommend using the following general rules address this question.

- In general a new GAMS variable that is specified to be positive should be defined for each identified potential infeasibility causing equation. This variable should have the same dimension as does the equation. Thus, if artificials are added to RESOUREQ(PLANT,RESOURCE), then the artificial should look like ARTRESOURQ(PLANT,RESOURCE).
- Artificials should be entered on the left-hand side of the equation with a coefficient of +1 in =L= equations and -1 in =G= equations. When the equation is an =E= the artificial should be a +1 if the equation right hand side is positive and -1 if it is negative.

Note, one cannot add an artificial into variable bounds which are defined using LO, .UP or .FX syntax. One needs to convert these to =G=, to =L= and =E= equations and then add the artificials following the above rules on signs.

One will also need to enter a large objective function penalty for the artificials. The coefficient will be negative when the objective function is maximized and when it is minimized. The magnitude of this penalty is entirely problem dependent and can cause numerical problems in the solver. All that can be said in general is that the penalty should dwarf the other objective function coefficients and should be large enough so that the artificial is driven to zero in any feasible model.

Alternatively, if numerical problems are plaguing the solution with the artificials entered, one can modify the objective function to one which minimizes some of the artificials. In the above example the simplest way to do this is to multiply the original objective function by zero and

convert the penalty on the artificial to something like hundred so that the shadow prices are not extremely small. This means the model becomes

```
variable           objmax;
positive variables x1, x2, A
equations          obj,r1,r2,r3;
obj..   objmax =e= 0*(50*x1 +50*x2)-100*A        ;
r1..                  x1 +   x2          =L=  50;
r2..               50*x1 +   x2          =L=  65;
r3..                  x1              +A =G=  20;
model infe /all/;
OPTION LP=BDMLP;
solve infe using lp maximizing objmax;
```

And yields a solution

```
**** OBJECTIVE VALUE            -1870.0000

                      LOWER     LEVEL     UPPER    MARGINAL
---- EQU obj            .         .         .        1.000
---- EQU r1           -INF       1.300    50.000       .
---- EQU r2           -INF      65.000    65.000     2.000
---- EQU r3          20.000     20.000     +INF    -100.000
                      LOWER     LEVEL     UPPER    MARGINAL
---- VAR objmax       -INF   -1870.000     +INF       .
---- VAR x1             .        1.300     +INF       .
---- VAR x2             .          .       +INF     -2.000
---- VAR A              .       18.700     +INF       .
```

Which again identifies the same infeasibility causing set.

3.1.3.4 How Are Distorted Marginals Identified?

The next question involves finding the distorted marginals. Under the BIG M method one reviews the output in the GAMS LST file reproduced above looking for marginals that have large absolute values or on our nonzero when working with the case just above where the original objective function was multiplied by zero. However, in models with thousands of variables and equations this information can be widely dispersed and difficult to find. The GAMSCHK procedure NONOPT can do this for you as when run it will automatically list out all items with marginals larger in absolute value than 10 to a filter value that is set through the Gams check option file (MARGFILT) for example adding the following to your code.

```
$onEOLcom
Modelname.optfile=1; //replace red part with the name of your mode)
*write solver option file for gamschk
File opt /gamschk.opt/;
Put opt
$onput
Margfilt 1
$offput
Putclose;
*write gck file telling gamschk what to do
File gck /%system.fn%.gck/;
Put gck
$onput
```

```
nonopt
$offput
Putclose;
*invoke gamschk as lp solver
Option LP=GAMSCHK;
```

3.1.4 Can you find what I did to the model?

If you want to play around with this a little bit here is a challenge. I made a version of the gams model library model Egypt (where I lengthened the abominably short choice of parameter and set names so the model was more easily comprehended). In that model I made a few subtle modifications that caused to be infeasible. I have also added in the needed code to add needed artificials (to get them you need to activate the set global statement in first line in the code) plus I added in stuff to start up gamschk (again needing activation by removing the * from column one in the line just before the word gamschk appears in the code). As a hint my modifications involve nutrition and land availability.

4 Integer and optimality

Another of the top 10 support questions involves solving an integer program and finding an answer that is inferior to a known one. This actually involves a couple of default parameter settings within GAMS that those running the company would really like to change but are hesitant to do so because of backwards compatibility. In particular the default upper bound on all integer variables is 100 and the optimality criteria setting (optcr) is set at 10%. This means if the best integer solution has a numerical value of an integer variable substantially above 100 or that the solution found is within 10% of the true optimum. This is fixed by placing

- new larger upper bounds on the integer variables or setting option intvarup to zero (which eliminates the bounds) and
- adding the statement option opcr=0.00001; or something similar to your code.

5 Courses offered

I will be teaching

- Basic to Advanced GAMS class Aug 7, 2017- Aug 11, 2017 (5 days) in the Colorado mountains at Frisco (near Breckenridge). The course spans from Basic topics to an Advanced GAMS class. Details are found at http://www.gams.com/courses/mccarl_combined.pdf
- Basic GAMS class Aug 7, 2017- Aug 9, 2017 (3 days) in the Colorado mountains at Frisco (near Breckenridge). The course starts assuming no GAMS background. Details are given at http://www.gams.com/courses/mccarl_basic.pdf.
- Advanced GAMS class Aug 9, 2017- Aug 11, 2017 (3 days) in the Colorado mountains at Frisco (near Breckenridge). The course is for users who have a GAMS background. Details are found at http://www.gams.com/courses/mccarl_advanced.pdf .

Further information and other courses are listed on http://www.gams.com/courses.htm . Note I also give custom courses for individual groups a couple of times a year.

6 Unsubscribe or subscribe to future issues of this newsletter

Please unsubscribe through the web form available at:
http://app.streamsend.com/public/XLmY/5eq/subscribe

Those who wish to subscribe to future issues can do this trough the newsletter section of http://www.gams.com/maillist/index.htm.

This newsletter is not a product of GAMS Corporation although it is distributed with their cooperation.

May 1, 2017