Bruce McCarl and Chengcheng Fei's GAMS Newsletter Number 47

February 2022

This Newsletter addresses the following

1 GAMS Releases in Last Year

Since the last newsletter we have new releases 35, 36, 37 and 38 with release notes being at https://www.gams.com/latest/docs/RN_MAIN.html. When we peruse these notes, we find a large number of bug fixes and allowable model size enhancements along with a few things that strike us to be of possible broad interest. These are:

1.1 GAMS system

- Added a new function **logit(x)** that computes the Logit Transformation: log(x/(1-x))
- Fixed references to external programs in **$Call, execute**, and their variants allowing spaces in a properly quoted string that gives the path of the script/program to be called.
- Updated the Python version used within GAMS to 3.8.12
- Added a new option command **dmpUserSym** to drop information that is generally not of use to users and dropped a number of irrelevant rows. They also added a column giving estimated memory use. The current display looks like the following for the AGRESTE model

```
SYMBOL TABLE DUMP (USER SYMBOLS ONLY), NR ENTRIES = 71
ENTRY                   ID   TYPE DIM  LENGTH MEMORYEST DEFINED ASSIGNED DATAKNOWN
  137                    c    SET   1       9       0 MB    TRUE    FALSE      TRUE
  138                    p    SET   1      12       0 MB    TRUE    FALSE      TRUE
  139                    s    SET   1       3       0 MB    TRUE    FALSE      TRUE
  140                   tm    SET   1      12       0 MB    TRUE    FALSE      TRUE
```

This provides information for finding issues with excessive memory use as discussed briefly in the GAMS user guide and more extensively in the McCarl guide.

- Added dollar control options **gdxLoad** and **gdxUnload** that permit one step loading and unloading of symbols from and to GDX files. While we doubt the usefulness of a compile time **gdxunload** but we feel the **gdxLoad** marginally simplifies things. The new command looks like

  ```
  $gdxLoad B.gdx d
  ```

  for loading the symbol d from the gdx file named B.gdx. The GDX file name specification does not carry over into subsequent statements

- Added a new command line parameter that altered the reference file pointers to ones that point to the beginning of the statement where an object appears rather than the actual line. Namely this happens when one uses **referenceLineNo=start** in association with a **rf=** command. For those unfamiliar with the reference file see coverage in my class notes on Using GAMS Studio.
- Improved LST file error messages associated with an execution error when one occurs during model generation (such as a division by zero error). The revised message now contains the equation name and index where the error occurs.
- Improved the ability to shorten the listing of include files in the LST file using the commands **$ONinclude** and **$OFFinclude**.
- Allowed access to values of some $ settings, environment variables and environment variables and command line parameters during a job. See the release notes for details.
- Introduced **put_utility** commands **setEnv** and **dropEnv** to allow users to change values of environment variables during execution.

1.2 Solvers

The release notes indicate that there were major releases with added new features for the solvers **BARON, KNITRO, SCIP (PaPILO , TBB, SoPlex), XPRESS, GUROBI, IPOPT, IPOPTH, CPLEX,** and **LINDO**. Additionally most other solvers had new releases with added features and/or bug fixes.

The notes also indicate that new solvers were added

- **COPT** – a parallel LP and MIP problem solver
- **OCTERACT** – a global optimization problem solver

Finally, the notes indicate that GAMS dropped the solvers **LocalSolver** and **LocalSolver70** and also announced plans to drop **Bonmin** and **BonminH** in a future release.

1.3 Studio

Studio has been augmented in a number of ways.

- The concept of a project as a collection of problem files have been embraced. However, this does not extend to all files in a directory but rather only to files that have been opened in Studio. We hope they further expand the concept and soon add a find option that searches all files in a file location. Because of that absence, we use GAMSIDE and will not be regular STUDIO users until such a search option (like grep or find in files or the equivalent) is implemented. We hear this is in the works perhaps to be included in the next release.
- Library files when opened are now placed into the currently active project directory provided the setting **open file in current project by default** is active.
- Added a search option to find text within a chosen selection.
- Added ability to specify the location of user defined libraries. For more on this see the presentation in the McCarl Guide at https://www.gams.com/mccarlGuide/using_another_model_library.htm
- Added code completion features some of which must be activated by **control-space**. Generally, we did not find the code completion to be very useful in most cases. Namely when testing this we at first got little more than added parenthesis or addition of a few characters after typing the beginning of a command like option. However, we later found after typing the word **option** that if one presses **control-space** one gets a useful drop down list of allowable options. Perhaps the code completion could be enhanced in other settings by adding a way to display an example for commands like sum and loop that identifies the generic structure like sum([setname(s)],[set_dependent_expression]);.
- Added ability to have STUDIO load GAMS execution created files whose names that are highlighted in the LOG file. This includes GDX, put and reference files and the loading is activated by a mouse click.

2 Scaling

After a discussion with GAMS staff on possible content that would be of interest to readers we decided to revisit scaling. Note the topic was covered before in newsletter 41. Here we cover some additional issues along with reinforcing some aspects. Namely, we address: 1) Why scaling may be needed based on computer numerical storage and calculation characteristics and things that happen with solvers; 2) How we figure out where to scale; and 3) Ways we use to develop scaling factors.

2.1 Why We Might Need to Scale?

Computers generally store numbers in binary form and fractional parts of numbers are not exactly represented unless they happen to be an even power of two. So a number like 1/3rd can never be exactly stored nor could we precisely store the result of 107/0.33333 both leading to what is commonly called round-off error. Furthermore, the imprecisions of such representations are compounded as more and more calculations are done. Additionally inside a computer, numbers are stored in terms of their exponent and their mantissa and when doing arithmetic operations like addition and subtraction, the exponents are synchronized and the representation of the mantissa can cause round-off error when the numbers are of substantially different magnitude (see Klotz 2014 for an extensive discussion[1]). Also conceptually significant errors can arise when doing operations such as simplex like pivots particularly when the computation involves large numbers being divided by smaller numbers.

The basic guidance arising here is keeping the magnitudes of the absolute values of the numbers in a similar range in a formulation helps avoid such problems. In addition, it is desirable to develop that range so the absolute values of the numbers surround one. While we have not found an explicit statement on that being a goal, it certainly motivates scaling procedures and discussions. For example Tomlin, 1975[2] mentions dividing rows through by the largest absolute value number; dividing through by the square root of the ratio between the largest and smallest absolute values and dividing through by the arithmetic mean. All of these end up with results with endpoints on or surrounding one. Furthermore, it is desirable to have the smallest possible range between the absolute values of the largest and smallest numbers targeting variation of say 3-5 orders of magnitude. It is also desirable to have gradients of nonlinear terms close to one, solution values of variables and shadow prices close to one as discussed by Drud in the CONOPT [3] manual. There is also mention of scaling by numbers that are powers of two which shifts the binary bits within the internal storage.

So all of that being said poorly scaled models containing a wide range between the absolute values of the largest and smallest numbers could cause round-off errors and result in less precise and possibly unreliable solutions. Such poor scaling can also degrade solver performance resulting in problems being incorrectly reported as infeasible, or falsely optimal along with solvers being stuck or making very slow progress. To avoid this or correct such issues it is often desirable to check scaling and, in turn, do some of your own scaling on the model or ask the solvers to employ more aggressive scaling.

[1] Klotz, Ed. "Identification, assessment, and correction of ill-conditioning and numerical instability in linear and integer programs." Bridging Data and Decisions. INFORMS, 2014. 54-108.
[2] Tomlin, John A. "On scaling linear programming problems." Computational practice in mathematical programming. Springer, Berlin, Heidelberg, 1975. 146-166.
[3] https://www.gams.com/38/docs/S_CONOPT.html?search=conopt%20scaling#CONOPT_SCALING

2.2 How Do You Determine Whether You Need to Scale?

Scaling is typically not a concern for small problems. But is larger problems with numbers varying widely solvers can

- refuse to solve the model due to disparity of numbers
- message issues if large number disparities are found
- report a condition number that is quite large
- call models infeasible after descaling
- not make much progress when solving
- terminate without reaching a solution

In such cases some attention to scaling may well be in order.

2.3 So How Might You Scale?

Three types of scaling can be undertaken: 1) solver scaling; 2) model formulation scaling; and 3) GAMS supported manual scaling.

2.3.1 Solver Scaling

Virtually all the GAMS solvers automatically and transparently scale problems. One can also invoke more aggressive scaling within a number of solvers through the option file. Ways of invoking such scaling are covered in the solver manuals.

2.3.2 Model Formulation Scaling

In setting up a model, a user chooses what units to use (tons, bushes, kilograms, 1000 tons etc.). Users can choose units in a manner that lowers the magnitude disparity in the model by preparing input data and expressing equations using units that lower the magnitude disparity between numbers and cause smaller values for solution variables and equation shadow prices (marginals). This involves for example setting up the objective function so it is in a unit like millions of dollars, expressing all RHS values in 1000s of units available and manipulating units so production and usage levels are represented by numbers smaller than 100. In agricultural models, we often put land and labor endowments in 1000's and commodity yields and sales in 100's. Naturally, such choices depend on the magnitude of the commodities being produced. In addition, we must admit that for model maintenance and consequent updating that we find it essential that we use units compatible with data sources. Nevertheless, we also find that direct use of such data frequently does result in well-scaled models. We address this by a) doing calculations to improve scaling in GAMS replacement statements (e.g. dividing things by 1000); b) by doing some scaling in model equation specifications (divide the objective function by 1000); and c) by using manual scaling as we discuss next.

2.3.3 GAMS Supported Manual Scaling

One can manually identify and use scaling factors using built-in scaling features in GAMS. This involves entering GAMS statements with variable and equation specific scaling amounts using commands like ***.scale*** and ***.scaleopt*** the basics of which are covered in the GAMS User Guide. For more extensively coverage and examples see newsletter 41, the scaling chapter of the McCarl Guide and from a theoretical standpoint chapter 17 of McCarl and Spreen.

How do we figure out the magnitude of scaling factors – Scaling implementation requires the identification of numbers (herein called scaling factors) used to divide every entry in an equation or multiply every entry associated with a variable. The size of such numbers is based upon the magnitude of the coefficients in the GAMS generated empirical model formulation. They are always problem context specific. A couple of general rules can be stated regarding identification of such scaling factors.

- It is desirable to examine the largest and smallest coefficients in absolute value for equations and variables and move them so they fall in a range that covers a value of one.
- There are limitations to simple scaling where one examines each variable or equation in isolation identifying scaling factors. In that case one sets the scaling factor to something like the mean absolute value causing after applying the scaling factor that the coefficients in that equation or associated with that variable to fall in a new range where the absolute values cover one. This will not generally gain advantage over solver scaling as that is what they will do possibly among other approaches.
- A highly effective way to lower magnitude disparity in model coefficients involves simultaneous/coordinated variable and equation scaling.
- Model structure knowledge can provide a basis for coordinated scaling by defining scaling factors for related variables and equations. For example
    - Suppose that within a model that a supply demand balance equation holds production plus imports plus storage withdrawals of a set of goods to be greater than or equal to sales plus exports plus addition to storage plus livestock feeding use. Such a case follows where variable and equation names are in upper case

      ```
      BALEQ(good)..
                    yld(good)*PROD +IMPORT(good)+STRWITHDRAW(good)
                    =l=
                    SALE(good)+EXPORT(good)+STRADD(good)+FEED(good);
      ```

      In such a case, it generally would be desirable to employ a common scale factor for both the balance equation as well as the import, export, storage, sales and livestock feeding variables as in the GAMS commands below.

      ```
      BALEQ.scale(good)              =scalevalue(good);
      IMPORT.scale(good)             =BALEQ.scale(good);
      STRWITHDRAW.scale(good)        =BALEQ.scale(good);
      ```

```
SALE.scale(good)          =BALEQ.scale(good);
EXPORT,scale(good)        =BALEQ.scale(good);
STRADD,scale(good)        =BALEQ.scale(good);
FEEDUSE.scale(good)       =BALEQ.scale(good);
```

  - Similar approaches can be used for resource equations. For example, labor equations would often involve use, labor hiring and family labor reservation where all those terms could be commonly scaled.
- It can be desirable to scale the objective function and equation right hand sides into 1000s of units to reduce solution value and shadow price magnitudes.
- In nonlinear models, scaling of nonlinear terms should generally involve an appropriate starting point as discussed in the McCarl guide in general and in reference to scaling.

2.4 So How Do We Arrive at Manual Scaling Factors?

Now we turn to the context specific question of how much should one scale. Here one needs to examine: a) the empirical model generated by GAMS during execution of a Solve statement; and/or b) any scaling information created by GAMS included "solvers". Note the word solver is placed in quotes since this can include GAMSCHK, which is not really a solver but does act like one and generates scaling related information.

An examination of the empirical model without solver help involves use of LIMROW/LIMCOL output. However, we feel this is an undesirable pathway due to the lack of targeted output. In particular, when using LIMROW/LIMCOL, one must visually scan or search for large and small exponents in often immense LST files. Beyond that

- One can use GAMSCHK (see the solver manual on this here) with first the BLOCKLIST or BLOCKPIC commands to find blocks of variables or equations with large or small numbers or consistent departures from values of one and then later MATCHIT to find individual cases. These GAMSCHK procedures give largest and smallest magnitudes of coefficient absolute values in equations and variables so you do not have to search as much. This can also be supported by use of DISPLAYCR.
- GAMS personnel told us for nonlinear cases it is useful to employ the CONVERT solver with the option 'jacobian'. We have never done this. This gives a gdx file with that contains a matrix of gradients of the nonlinear terms (evaluated at the starting point). Then one can export that file to Excel and examine the results. Note when doing this CONVERT replaces all the internal variable and equation names. Subsequently the variables are named x1,x2,x3,... and the equations e1,e2,e3,...> To get back to the original names one needs to manually back translate using the CONVERT generated dictionary file. We have no experience with this but it just does not seem practical in large models as the GDX and Excel might be unwieldy as would be the backward translation.
- GAMS personnel also told us about an option in CPLEX (datacheck=2) that reports suspicious item in a model. In turn the log file (not the LST file) contains information

about large and small coefficients, equation RHS values and anything else Cplex thinks can cause numerical instability. We tried to use this recently on a large model but found it did not help and even led us down some improper paths and finally abandoned it.

Across these, not surprisingly since McCarl wrote it, we prefer the GAMSCHK approach where

- BLOCKLIST tells you the size of the largest and smallest numbers in absolute value within each variable and equation block. BLOCKPIC also does this and simultaneously identifies the largest and smallest numbers at variable/equation block intersections.
- MATCHIT tells you the size of the largest and smallest numbers in absolute value for **requested** variables and equations.
- DISPLAYCR allows one to look at coefficients within specific variables and equations to form exact scaling values.
- PICTURE allows one to look at magnitudes of numbers in the model at intersections of variables and equations. Note in use of PICTURE, one can restrict attention to matrix components by using intersection mode and naming specific variables and equations.
- Caution is needed with nonlinear models as the coefficients reported in GAMSCHK (and LIMROW/LIMCOL) depend on starting point and thus a good starting point is needed.

2.5 Example

Since scaling is context specific so is the exact approach one would follow to determine appropriate scaling factors. To provide some insight a GAMSCHK supported scaling exercise is carried out on the AGRESTE model from the GAMS model library. Note that this model does not need scaling to be successfully solved but it does provide the basis for a scaling exercise.

The first step we follow involves use of GAMSCHK with the option BLOCKLIST. This is done by adding the following code to the AGRESTE file before the first solve. Note we also commented out the second solve.

```
option lp=gamschk;
file gck /%system.fn%.gck/;
put gck;
$onput
blocklist
$offput
putclose;
```

After adding this a run of the modified model resulted in the following material that was extracted from the LST file

Variable Block	Sign Res	Numb Vars	Numb Nonl	Pos Coef	Neg Coef	Nonl Coef	Maximum Absolute	Minimum Absolute
xcrop	>=0	23	0	332	58	0	291.0	0.3000E-01
xliver	>=0	3	0	42	6	0	49.84	0.4000E-01
xlive	<0>	1	0	1	2	0	211.0	1.000
lswitch	>=0	2	0	2	2	0	1.000	1.000
xprod	<0>	9	0	9	0	0	1.000	1.000
cons	>=0	3	0	3	12	0	2.640	0.1500
sales	>=0	9	0	21	47	0	0.3500E+05	0.5357
flab	>=0	12	0	0	24	0	3.000	1.000
tlab	>=0	12	0	12	24	0	10.00	1.000

```
plab            >=0     1     0     1    13     0     2054.       25.00
rationr         <0>     1     0     3     0     0     1.000       1.000
pdev            >=0    10     0    10    10     0     1.000       1.000
ndev            >=0    10     0    20     0     0     1.000       1.000
yfarm           <0>     1     0     1     0     0     0.000       0.000
revenue         <0>     1     0     1     1     0     1.000       1.000
cropcost        <0>     1     0     3     0     0     1.000       1.000
labcost         <0>     1     0     2     0     0     1.000       1.000
vetcost         <0>     1     0     3     0     0     1.000       1.000


----### List of Equation Block Characteristics

Note Max and Min do not include RHS and Objective variable

Equation         Type  Numb Numb Pos  Neg  Nonl Pos  Neg  Maximum    Minimum
  Block          Res   Eqns Nonl Coef Coef Coef RHS  RHS  Absolute   Absolute
landb            =L=     3    0   25    2    0    3    0   2.030      0.2090
lbal             =E=     1    0    1    3    0    0    0   1.000       1.000
rliv             =E=     1    0    1    3    0    0    0   49.84       1.000
mbalc            =G=     9    0   39   18    0    0    0   4.456      0.3000E-01
dprod            =E=     9    0    9   39    0    0    0   4.456      0.3000E-01
cond             =E=     1    0    3    0    0    1    0   1.000       1.000
labc             =L=    12    0  312   36    0    0    0   28.12      0.4000E-01
ddev             =E=    10    0   31   39    0    0    0   7757.      0.5357
income           =E=     1    0   25    4    0    0    0   934.0      0.1000
arev             =E=     1    0    1   10    0    0    0  0.3500E+05  1.000
acrop            =E=     1    0    1   19    0    0    0   291.0       1.000
alab             =E=     1    0    1   25    0    0    0   2054.       1.000
awcc             =L=     1    0   16    0    0    1    0   2054.       1.000
avet             =E=     1    0    1    1    0    0    0   1.000       1.000
```

Here there are a number of things that could be scaled but we chose to initially focus on the small numbers in the xcrop variable which appear in the mbalc row because it allows an example of coordinated scaling. Consequently we would then add the following entries to the GAMSCHK input file

```
displaycr
      equations
      mbalc
```

where we could also have included MATCHIT. The resultant LST file contains entries like that is just below where we are only including equations for three of the commodities to save space.

```
----## EQU mbalc
 ##  mbalc(cotton-h)
 xcrop(crop-02,good)                                                0.84800
 xcrop(crop-02,medium)                                              0.56900
 xcrop(crop-29,good)                                                0.26900
 xcrop(crop-29,medium)                                              0.14900
 xcrop(crop-30,good)                                                0.40300
 xcrop(crop-30,medium)                                              0.13300
 sales(cotton-h)                                                   -1.0000
        =G=                                                         0.0000

 ##  mbalc(oranges)
 xcrop(crop-15,good)                                               0.92000E-01
 sales(oranges)                                                    -1.0000
        =G=                                                         0.0000

 ##  mbalc(sisal)
 xcrop(crop-19,good)                                                2.2440
 xcrop(crop-19,medium)                                              1.6660
 sales(sisal)                                                      -1.0000

        =G=                                                         0.0000
```

In turn, we would examine the coefficients within these equations and choose scaling factors such that they resulted in magnitudes of coefficients within those equations that have a range that includes one. We would also do a coordinated scale where we would simultaneously scale the sales variable by the same amount. The consequent scaling implementation we entered into GAMS follows

```
agreste.scaleopt=1;
mbalc.scale("cotton-h  ")=1/4;
mbalc.scale("banana    ")=1/5;
mbalc.scale("sugar-cane")=1/25;
mbalc.scale("beans-arr ")=1/4;
mbalc.scale("beans-cor ")=1/5;
mbalc.scale("oranges   ")=0.092;
mbalc.scale("manioc    ")=1/1;
mbalc.scale("corn      ")=1/2;
mbalc.scale("sisal     ")=2;
sales.scale(c)=mbalc.scale(c);
```

Note that in general when equation names are specified that GAMS divides every coefficient for each specified equation through by the scale factor. For variables, every coefficient associated with that variable is multiplied by that scale factor.

The resultant DISPLAYCR shows

```
----## EQU mbalc
 ##  mbalc(cotton-h)
 xcrop(crop-02,good)                                                 3.3920
 xcrop(crop-02,medium)                                               2.2760
 xcrop(crop-29,good)                                                 1.0760
 xcrop(crop-29,medium)                                              0.59600
 xcrop(crop-30,good)                                                 1.6120
 xcrop(crop-30,medium)                                              0.53200
 sales(cotton-h)                                                   -1.0000
         =G=                                                        0.0000

 ##  mbalc(oranges)
 xcrop(crop-15,good)                                                 1.0000
 sales(oranges)                                                    -1.0000
         =G=                                                        0.0000

 ##  mbalc(sisal)
 xcrop(crop-19,good)                                                 1.1220
 xcrop(crop-19,medium)                                              0.83300
 sales(sisal)                                                      -1.0000
        =G=                                                         0.0000
```

where the coordinated simultaneous scaling factor definitions for Mbalc and sales cause division of the **mbalc** equation coefficients and multiplication of the **sales** variable associated coefficients. Further in this case the coordinated multiplication and division of the coefficient for the sales variable in the mbalc row by the same scaling factor left the sales coefficients in those equations at a value of one and reduced the disparity of the numbers across the equation.

A number of other scaling manipulations were done yielding the following manual scale instructions that were included in the AGRESTE model.

```
agreste.scaleopt=1;
mbalc.scale("cotton-h  ")=1/4;
mbalc.scale("banana    ")=1/5;
mbalc.scale("sugar-cane")=1/25;
mbalc.scale("beans-arr ")=1/4;
mbalc.scale("beans-cor ")=1/5;
mbalc.scale("oranges   ")=0.092;
mbalc.scale("manioc    ")=1/1;
mbalc.scale("corn      ")=1/2;
mbalc.scale("sisal     ")=2;
sales.scale(c)=mbalc.scale(c);
dprod.scale(c)=mbalc.scale(c);
arev.scale=500;
revenue.scale=arev.scale;
yfarm.scale=arev.scale/100;
income.scale=arev.scale/100;
ddev.scale(ty)=10;
pdev.scale(ty)=ddev.scale(ty);
ndev.scale(ty)=ddev.scale(ty);
plab.scale=1/100;
xliver.scale(r)=10;
rliv.scale=10;
rationr.scale=rliv.scale;
acrop.scale=100;
cropcost.scale=acrop.scale;
awcc.scale=10;
vetcost.scale=awcc.scale;
labcost.scale=10;
alab.scale=10;
```

The resulting model after scaling will contained a largest coefficient of 186 and a smallest of 0.054 as opposed to the before scaling range from 35,000 to 0.03. No meaningful changes in the solution were observed with this scaling and the scaled version actually took one more iteration to solve. So again small models do not generally require scaling actions but unfortunately our experience shows big ones do.

2.6 Use of Manual Scaling Does Not Alter the Reported Solution

The last point worth making is that this manual scaling does not affect the magnitudes of the numbers in the resulting solution. GAMS uses a known exact backwards transformation that yields the unscaled solution for use in reporting. For those interested in details on the backwards transformation a presentation on that appears in McCarl and Spreen chapter 17. Users can also test this by running with and without the **scaleopt** feature set to one and then observing that the GAMS reported solution does not change.

3 GAMSCHK POSTOPT Bug Fixed But Another Popped Up

For a while there have been cases when POSTOPT gave some bad results. We finally found a case that consistently malfunctioned and worked with GAMS staff to get the FORTRAN code fixed. In particular, we found when the NOSOLVE option was being used then there were some uninitialized memory locations leading to bad reporting of the solution information. That was fixed and the fix was included in last week's GAMS 38 version 2 maintenance release. However, Uwe Schneider quickly found out that POSTOPT now was reporting faulty variable levels and equation marginals. In turn, we figured out that solving a model twice (with a second

Solve statement in the source code) was a work around. GAMS staff have now fixed the problem but wont send it out until the next maintenance release. So for now users should either use versions earlier than the very last maintenance release (38.2) without using the NOSOLVE option or solve twice with GAMSCHK used in the second solve.

4 GAMS Transfer

GAMS has recently announced the creation of a product it calls TRANSFER. The blog announcement indicates TRANSFER is designed to simplify the process of exchanging data with other programs. Initially they have focused on transfers with Python and Matlab. The blog announcement indicates that TRANSFER development focused on:

- **Speed**: Fast Performance in the transfer of large datasets.
- **Convenience**: creation of a facility that is intuitive to use and compatible with target program specific data formats.
- **Consistency**: to the extent possible the development of a common syntax that applies across different environments (target programs and computing environments).

An introduction to features of GAMS Transfer was presented at the 2021 Informs Annual Meeting. This is available at https://www.gams.com/blog/2021/11/informs-annual-meeting-in-anaheim/assets/GAMS_Transfer_INFORMS2021.pdf. Also an example of the use of TRANSFER is presented at the GAMS webpage https://www.gams.com/blog/2021/11/introducing-gams-transfer/

5 GAMS Engine

GAMS Engine allows scheduling and running jobs on a central computing server or now in the Amazon Web Services cloud. According to a recent GAMS blog entry it is designed to ease the burden and cost of developing and integrating GAMS based modeling into IT environments.

It's use involves, from what we can tell as a non-user, the use of Python to generate multiple GAMS jobs and integrate the solutions. It requires a single overall GAMS Engine license that appears to be set up between the user and GAMS. More can be found on this on the GAMS Blog at https://www.gams.com/blog/2022/01/introducing-engine-saas/ .

6 Basic, Advanced and Combined Courses Offered Soon

This year we will again be teaching my family of GAMS courses for basic and advanced users. These courses will be offered in May again via ZOOM due to the continuing pandemic. Dates, times and content are

- Basic to Advanced GAMS class May 9 - May 13 and May 16 (six 5 1/2 hour sessions each 8-1:30 central). The course spans from Basic topics to an Advanced GAMS class. Details are found at https://www.gams.com/courses/2022_05_basic_to_advanced-gams-modeling_mccarl/

- Basic GAMS class May 9 - May 12 (four 5 1/2 hour sessions each 8-1:30 central) The course starts assuming no GAMS background. Details are found at https://www.gams.com/courses/2022_05_basic-gams-modeling_mccarl/
- Advanced GAMS class May 11 - May 13 and May 16 (four 5 1/2 hour sessions each 8-1:30 central). The course is for users who have a GAMS background. Details are found at https://www.gams.com/courses/2022_05_advanced-gams-modeling_mccarl/ Note we also give custom courses for individual groups a couple of times a year.

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

7 Unsubscribe or Subscribe to Future Issues of this Newsletter

Please unsubscribe through the web form available at: https://gams.us18.list-manage.com/unsubscribe?u=f1497c76718404eae593beb11&id=45ccea2ee0

Those who wish to subscribe to future issues can do this through 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.

February 23, 2022