Bruce McCarl's GAMS Newsletter Number 29

Here I

- Announce a new documentation version
- Review developments in releases GAMS 23.4 and 23.5 including putting links in spreadsheets when using GDXXRW
- Include a contribution from Tom Rutherford on Spreadsheets and Pivot tables
- Mention an upcoming future course

Expanded GAMS User Guide by McCarl et al.

I updated the User's Guide to reflect 23.4 and 23.5 with changes added here and there. This may be at
http://www.gams.com/dd/docs/bigdocs/gams2002/mccarlgamsuserguide.pdf and will be in upcoming GAMS releases.

GAMS Features in releases 23.4 and 23.5

New language elements

- A new model attribute Trylinear has been included that if activated (modelname.trylinear=1;) causes empirical NLP, MINLP and RMINLP (plus QP variants) models to be checked for NLP terms and if none are present solved using the default LP, MIP or RMIP solver.
- A new compile time usage of sameas has been added so one can use it in $eval (but not in $if) as in the following code

```
$setglobal a doit
$eval a sameas(%tt%,"doit")
$if "%a%"=="1" x=x*4;
```

- A set of new constants has been added that one can enter into the code to test of optimality status, grid problem solution status and other things. These involve a family associated with:

  Solprint involving solution printing with constants Solprint.Summary, Solprint.Report and Solprint.Quiet.

  HandleStatus involving grid computing problem status with constants Handlestatus.Unknown, Handlestatus.Running, Handlestatus.Ready, and Handlestatus.Failure.

  Solvelink involving in core solver links and grid computing with constants Solvelink.ChainScript, Solvelink.CallScript, Solvelink.CallModule, Solvelink.AsyncGrid, Solvelink.AsyncSimulate, and Solvelink.LoadLibrary.

  Solvestat involving solver completion status with constants: Solvestat.Normal Completion, Solvestat.Iteration Interrupt, Solvestat.Resource Interrupt, Solvestat.Terminated By Solver, Solvestat.Evaluation Interrupt, Solvestat.Capability Problems, Solvestat.Licensing Problems, Solvestat.User

Interrupt, Solvestat.Setup Failure, Solvestat.Solver Failure, Solvestat.Internal Solver Failure, Solvestat.Solve Processing Skipped and Solvestat.System Failure

Modelstat involving model optimality status with constants: ModelStat.Optimal, ModelStat.Locally Optimal, ModelStat.Unbounded, ModelStat.Infeasible, ModelStat.Locally Infeasible, ModelStat.Intermediate Infeasible, ModelStat.Intermediate Nonoptimal, ModelStat.Integer Solution, ModelStat.Intermediate Non-Integer, ModelStat.Integer Infeasible, ModelStat.Licensing Problems, ModelStat.Error Unknown, ModelStat.Error No Solution, ModelStat.No Solution Returned, ModelStat.Solved Unique, ModelStat.Solved, ModelStat.Solved Singular, ModelStat.Unbounded - No Solution, ModelStat.Infeasible - No Solution,

These are predefined constants and cannot have their values changed but can be used in statements like

```
Modelname.solprint=% Solprint.Quiet%;
If(modelname.modelstat le %ModelStat.Locally Optimal%,
        display ‘optimal solution found’);
```

GDXXRW Features in releases 23.4 and 23.5

GDXXRW was modified to support

- Writing an Excel file with filters (see details here)
- Writing text and hyperlinks into a spreadsheet.

The text and hyperlink features are illustrated by

```
execute 'gdxxrw gdxrw.gdx o=gdxxrwss.xls text="This is the link dictionary" rng=linkdictionary!a1  '
execute 'gdxxrw gdxrw.gdx o=gdxxrwss.xls text="This is the twodim parameter" rng=output!a1  '
execute 'gdxxrw gdxrw.gdx o=gdxxrwss.xls par=twodim rng=output!b2 text="Link to twodim" rng=linkd!a2 linkid=twodim  '
execute 'gdxxrw gdxrw.gdx o=gdxxrwss.xls text="This is the threedim parameter" rng=output!a10  '
execute 'gdxxrw gdxrw.gdx o=gdxxrwss.xls par=threedim rng=output!a12 text="Link to threedim" rng=linkd!a3 linkid=threedim'
execute 'gdxxrw gdxxrwwrite.gdx o=gdxxrwss.xls text="Link to threedim second variant" rng=linkd!a4 link=output!a33'
```

Solver developments in releases 23.4 and 23.5

A number of solver developments occurred with

- New libraries for BARON, CPLEX, GUROBI, LINDOGLOBAL, MOSEK and XPRESS plus the COIN solvers CBC, Bonmin, Couenne, Ipopt, GLPK, Mumps and OS
- The coin solvers were renamed dropping COIN from the name of BONMIN, CBC, COUENNE, GLPK, IPOPT, OS, and SCIP.
- The bare bone versions of the commercial solvers were renamed to OSICPLEX, OSIGUROBI, OSIMOSEK and OSIXPRESS.
- Substantial features were included in the new CPLEX version involving MIPs, and concurrent solves
- Substantial features were included in the new GUROBI version involving a parallel barrier solver, MIP features, presolves, crossovers and multi core computing.

Other features

- mdb2gms, sql2gms and gdx2access now support saving the text associated with set entries in a database.
- mdb2gms has a parameter to choose the database type (.mdb or .accdb)
- A new model type `EMP` has been added for extended mathematical programming. Several extensions and examples are provided. See discussion in the release notes.
- The EMP solver was renamed to JAMS and is the default solver for EMP models.

A contribution from Tom Rutherford

Tom Rutherford provided a piece for inclusion in the newsletter that is below after a little editing

Features that facilitate movement of data to EXCEL Pivot Tables by Tom Rutherford

Excel provides a functional framework for presenting GAMS model results. Used properly, an Excel report file can help a model user investigate the consequences of policy experiments. The most helpful tool provided within Excel for this purpose is the Pivot Table. I have found that Pivot Tables provide a productive framework for communicating GAMS model results to interested but less modeling adept clients.

There are a few GAMS programming "tricks" which facilitate the transformation of GAMS model outputs into Excel pivot tables. These include:

- The $echov statement as a means of generating "batinclude subroutines".
- The $macro statement as a means of improving compactness and precision of GAMS code for computing model reports.
- The DOS FOR statement with command extensions for transferring model output to all Excel files in a folder.

Tom supplied a little GAMS program, which illustrates how these various programming features can be applied that is called tompivot.gms.

In explaining this code Tom indicates

- He uses the XLSX Excel file format - the standard format for Excel 2007 as there is a much larger upper bound on the number of rows in an individual worksheet (~1.5 million) which helps for pivot report tables with many keys or lots of scenarios.

- The usefulness of the "for %F in (xls*.xlsx) do ()" DOS statement may not be immediately evident. This statement applies the same GDXXRW data transfer to every worksheet in the xls directory which is very helpful if you have worked with the pivot report and produced one or more report tables or charts. This syntax assures that you can update all of the report files related to your model automatically.

- An important advantage of this approach is that if you discover a glitch in your model and find that you need to rerun all the cases, then you won't need to lose your earlier edits. You can work with Excel and pivot any number of reports, save these in any number of files in the report directory and then having these reloaded each time you rerun your model.

He also points out some issues with GDXXRW use

- Advanced Excel users may be aware that you can set a Pivot Table switch requesting that a pivot table cache be automatically reloaded each time the workbook opens. Alas, if you set this option in Excel, GDXXRW crashes when trying to move data into the Workbook. Who knows, perhaps this can be fixed in future versions of GDXXRW?
- There is a major annoyance related to how GDXXRW transfers integer labels. In order to have the pivot table sort rows numerically, you need to convert the labels in column A of the PivotData worksheet to numeric format. Click on the column heading (A) while the yellow box is visible and while holding down the shift key. Then click "convert to number". This is a real pain. Perhaps someone can tell us how to do this automatically?

Courses offered

I will be teaching

- Advanced GAMS class Aug 10-13, 2010 (3 1/2 days) in the Colorado mountains at Frisco (near Breckenridge). The course covers such diverse topics as links to other programs like macros, spreadsheets, speeding up GAMS, scaling, debugging, improving output and advanced basis use along with many other topics. It also covers the techniques and principles used in a large US model FASOM.
- Further information and other courses are listed on http://www.gams.com/courses.htm.

Unsubscribe to future issues of this newsletter

To remove your name, please send an email to mccarl-news-request@gams.com containing unsubscribe on the subject line or unsubscribe through the web form http://www.gams.com/maillist/newsletter.htm.

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

July 24, 2010

```
$title        Little GAMS Program from Tom Rutherford that Illustrates Report Generation
with Excel

*        Use the canonical transport model as illustration.   We begin by using
*        GAMSLIB to retrieve a copy in the current directory:
$call 'gamslib trnsport'
*        Include the library model.  This defines the model and
*        provides an initial solve:
$include trnsport
*        Create a parameter in which to store the reference solution.
*        We use this solution as a reference point from which to
*        evaluate changes induced by policy intervention:
parameter        x0(i,j)        Benchmark trade flows;
x0(i,j) = x.l(i,j);

*        Declare some scenarios to compute:
set        scn /0,50,100,150,200,250,300/;

*        Define a macro, which computes percentage changes:

$macro        pct(x0,xscn)        ((100*(xscn/x0-1))$(x0>0))

*        Add the following if you want to label changes from
*        a zero base:

*                (+inf)$(x0=0 and xscn>0) + (-inf)$(x0=0 and xscn<0))

*        Declare a parameter in which to store model results.
*        In a more complicated model there could be many such
*        parameters.

parameter        report        Summary report;

*        Generate a "reporting subroutine".
*        The $echov syntax permits us to write %1 in the
*        file without having it expanded.

*        We save the subroutine in the scratch directory
*        with an ".scr" suffix so that it is erased at the
*        end of the GAMS job.


$onechov >%gams.scrdir%report.scr
report(%1,"lvl",i,j) = x.l(i,j);
report(%1,"%",i,j)   = pct(x0(i,j),x.l(i,j));
$offecho

*        Loop over scenarios with computations and reporting.

*        More complicated models might be better processed one
*        by one with output saved to individual .gdx files.
option solprint=off; option limrow=0; option limcol=0;

loop(scn,

*        Assign the scenario policy -- an upper bound on
*        flows from Seattle to Chicago:

        X.UP("seattle","chicago") = scn.val;

        Solve transport using lp minimizing z ;

*        Use the report code as a subroutine:

$batinclude %gams.scrdir%report.scr scn

);

*        Unload scenario results in a GDX file:

execute_unload 'pivotdata.gdx',report;
```

```
*        When an xls output directory already exists, move the
*        pivot report data into every XLSX file in that directory:

$ifthen exist '.\xls\nul'
  execute 'for %F in (.\xls\*.xlsx) do (call gdxxrw i=pivotdata.gdx o=.\xls\%~nF.xlsx
par=report rng=PivotData!a2 cdim=0)';

$else

*        If no xls output directory exists, create one and dump
*        report data into a new report file there:

  execute 'mkdir .\xls';
  execute 'gdxxrw i=pivotdata.gdx o=.\xls\report.xlsx par=report rng=PivotData!a2
cdim=0';
$endif
```