```
$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


$ontext
I use the XLSX Excel file format here which is the standard format for
Excel 2007.  An important advantage of using this format is that there
is a much larger upper bound on the number of rows in an individual
worksheet (~1.5 million).  This helps a lot 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.

PS. 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?
```

```
PPS.  There is a major annoyance related to how GDXXRW transfers
integer labels.  In orrder 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 headding
(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?
$offtext
```