Examples of using
Replaceable parameters in CMF files

Mark Horridge and Michael Jerie, May 2013

Starting with Release 11.2, GEMPACK allows you to pass replaceable parameters from the command line to CMF files. For example, either of the commands:

```
gemsim  -cmf sim.cmf -p1="1.3"
mymodel -cmf sim.cmf -p1="1.3"
```

will cause each occurrence of <p1> within sim.cmf to be replaced at runtime by 1.3.

For example, if sim.cmf contained the line:

```
shock   realwage = <p1> ;
```

GEMPACK would translate this at runtime to:

```
shock   realwage = 1.3 ;
```

The facility is designed to assist in the running of multiple similar simulations using BATch files. The syntax and operation mimics the operation of the replaceable parameters %1, %2, and so on, which can appear in a BAT file.

More details appear in the GEMPACK Manual, Section 20.9.[1]

This document accompanies a zip archive containing several examples showing how the facility might be used. The archive can be downloaded from:

http://www.monash.edu.au/policy/archivep.htm#tpmh0129

In the examples which follow it is assumed that you have unzipped tpmh0129.zip into a new folder, and that you have opened a command prompt [aka DOS box] in that folder.

The archive includes compiled model files [TERM.exe, TAXLOSS.exe] compiled with GEMPACK Release 11.2. You can run the examples with this model even if your GEMPACK licence is older than this.

Example 1: Specifying the shock from the command line

To try the first example, type

```
term -cmf basic1.cmf -p1=10
```

A simulation should run, producing solution file basic1.sl4. Now examine the command file basic1.cmf.

You will see that the shock statement is:

```
shock pfimp("machines",ORG) = uniform <p1>;
```

At runtime the "<p1>" is replaced by the command line parameter "10".

You can easily run more simulations with different shocks by typing:

```
term -cmf basic1.cmf -p1=20
term -cmf basic1.cmf -p1=30
term -cmf basic1.cmf -p1=40
         and so on
```

A limitation is that each simulation produces the same solution file basic1.sl4. That problem is addressed in the next example.

[1] Or look in the index under "Command file: command-line parameters" or " Parameters: passed to CMF files "

Example 2: Specifying shock and SL4 file from the command line

To try this example, type

```
term -cmf basic2.cmf -p1=10 -p2=sim1
```

A simulation should run, producing solution file sim1-10.sl4. Now examine the command file basic2.cmf. You will see that we added there the line:

```
solution file = <p2>-<p1>;
```

At runtime the "<p1>" is replaced by the command line parameter "10", and the "<p2>" is replaced by the command line parameter "sim1".

You can run more simulations with different shocks by typing:

```
term -cmf basic2.cmf -p2=sim1 -p1=20
term -cmf basic2.cmf -p2=sim2 -p1=30
term -cmf basic2.cmf -p2=sim3 -p1=40
         and so on
```

to produce files sim1-20.sl4, sim2-30.sl4, and sim3-40.sl4.

A BAT file could be used to automate the above commands. Type

```
basic2.bat
```

to see such a BAT file run. You can examine basic2.bat in TABmate to see:

```
del sim*.sl4
term -cmf basic2.cmf -p2=sim1 -p1=20 >nul
term -cmf basic2.cmf -p2=sim2 -p1=30 >nul
term -cmf basic2.cmf -p2=sim3 -p1=40 >nul
dir sim*.sl4
```

The ">nul" in the lines above prevents output from cluttering the screen.

A similar effect is achieved by typing:

```
basic2a.bat
```

which has lines:

```
rem Example using the batch FOR command
del shk*.sl4
FOR  %%a IN (10,20,30,40,50) DO (
term -cmf basic2.cmf -p1=%%a -p2=shk >nul
)
dir shk*.sl4
rem see variable "NatMacro" in ViewSOL
START viewsol shk*.sl4
```

Basic2a.bat uses two more advanced BAT file tricks:

- the FOR command to successively set %%a to 10,20,30,40,50
- the START command to view the solutions in ViewSOL

The web page

```
http://www.monash.edu.au/policy/gp-bat.htm
```

contains a basic introduction to BAT file use with GEMPACK

Example 3: Testing different solution methods for speed

You may have heard rumours that obscure CMF file options, like:

```
LU decompose transpose = yes ;      ! default is NO
```

may in some case speed up your simulation. The next example tests all possible combinations of:

- IZ1 = YES/no;

- LU decompose transpose = yes/NO;
- Markowitz pivots in MA48 = yes/NO;
- m28 = yes/NO;

to see which combination solves quickest (in each case default option is shown uppercase). Type:

```
solmethod.bat
```

to see it run. After a while, you should see something like:

```
SOL_IZ1N_TRNN_MARN_M28N.log:  [Total CPU is 6.5832419 seconds.]
SOL_IZ1N_TRNN_MARN_M28Y.log:  [Total CPU is 6.8484440 seconds.]
SOL_IZ1N_TRNN_MARY_M28N.log:  [Total CPU is 10.093265 seconds.]
SOL_IZ1N_TRNN_MARY_M28Y.log:  [Total CPU is 6.8796439 seconds.]
SOL_IZ1N_TRNY_MARN_M28N.log:  [Total CPU is 11.622074 seconds.]
SOL_IZ1N_TRNY_MARY_M28N.log:  [Total CPU is 6.3492408 seconds.]
SOL_IZ1Y_TRNN_MARN_M28N.log:  [Total CPU is 6.5676422 seconds.]
SOL_IZ1Y_TRNN_MARN_M28Y.log:  [Total CPU is 21.918140 seconds.]
SOL_IZ1Y_TRNN_MARY_M28N.log:  [Total CPU is 16.692106 seconds.]
SOL_IZ1Y_TRNN_MARY_M28Y.log:  [Total CPU is 21.637339 seconds.]
SOL_IZ1Y_TRNY_MARN_M28N.log:  [Total CPU is 12.308478 seconds.]
SOL_IZ1Y_TRNY_MARY_M28N.log:  [Total CPU is 9.0480576 seconds.]
```

showing solution times (really only accurate to nearest half-second) for the different combinations[2]. The default option (marked bold) seems as good as any in this case. The batch file SolMethod.bat has lines:

```
echo off
del *.log
FOR  %%a IN (Y,N) DO (
FOR  %%b IN (Y,N) DO (
FOR  %%c IN (Y,N) DO (
FOR  %%d IN (Y,N) DO (
echo computing with IZ1=%%a  LUtranspose=%%b  MarkowitzPivots=%%c  M28=%%d
term -cmf solmethod.cmf -p1=%%a -p2=%%b -p3=%%c -p4=%%d >nul
if errorlevel 1 echo Invalid combination of options: LUtranspose AND M28
)
)
)
)
echo Solution times were:
findstr /I /C:"Total CPU" sol*.log
rem LU decompose transpose = yes ; only works with MA48 (not with MA28)
rem Markowitz pivots applies only to MA48; does not affect MA28
```

Points to note above are:

- 4 nested FOR loops specify the 16 possible combinations
- The ECHO command controls which output is echoed to screen
- The FindStr command extracts Total CPU times from the log files

The first 5 lines of SolMethod.CMF are:

```
IZ1 = <p1> ;
LU decompose transpose = <p2>
Markowitz pivots in MA48 = <p3>
m28 = <p4> ;   ! .
log file = SOL_IZ1<p1>_TRN<p2>_MAR<p3>_M28<p4>.log;
```

[2] Not all 16 combinations are valid, so LOG files of 4 failed runs do not appear (hint: CMF log file statement came AFTER option choices).

Example 4: Shocks from a file, and graphing solution values in Excel

Type:

```
term -cmf ImpPrice.cmf -p1=10
```

to run a simulation producing ImpPrice.sl4. The first line of ImpPrice.cmf is:

```
shock pfimp("OthManufact",ORG) = uniform <p1>;
```

so that this simulation raises the price of foreign OthManufact by 10%. Now type:

```
ImpPrice.bat
```

and you should see a series of simulations running, with steadily increasing shocks. The BAT file is:

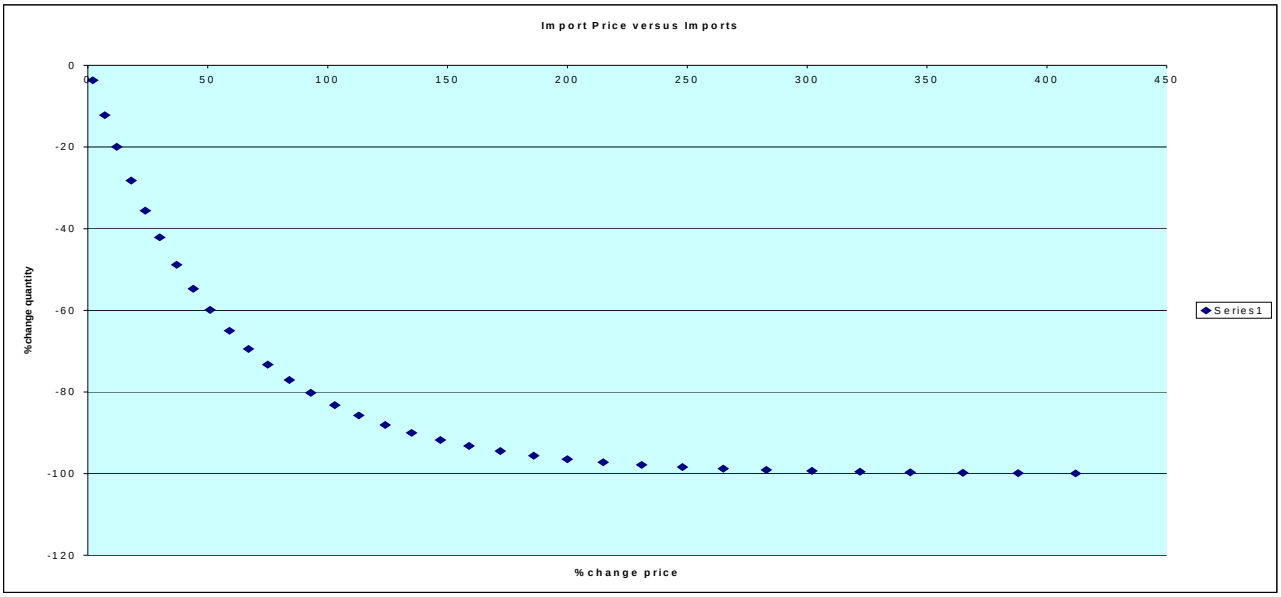
```
echo off
del imp*.csv
FOR /F  %%a IN (impprice.shk) DO (
echo computing with shock=%%a
term -cmf impprice.cmf -p1=%%a  >nul
if errorlevel 1 echo Simulation failed
sltoht -sss   -map=impprice.map impprice.sl4  impprice%%a.csv  >nul
)
echo results in  impgraph.csv
findstr /I /C:"java" imp*.csv >impgraph.csv
```

Points to note above are:

- The FOR /F command draws successive values of %%a from each line of the file ImpPrice.shk (examine this file in TABmate). Each time, it executes the lines between the opening parenthesis (after DO) and the closing parenthesis (after the sltoht line).
- The Sltoht command extracts two values from ImpPrice.sl4, and writes them to a uniquely named CSV file. ImpPrice.map is a *spreadsheet mapping file* with just one line:

  ```
  pfimp("OthManufact","Java") : ximps
  ```

- The FindStr command extracts relevant lines from the different CSV files made by Sltoht, and stores them in the file ImpGraph.csv (which you should inspect with TABmate).
- Now a pretty graph ! Start Excel, and first, open ImpGraph.csv. *Second*, open ImpGraph.xls -- which is "linked" to ImpGraph.csv. You should see a graph like the below, relating (%changes in) ximps("OthManufact","Java") to pfimp("OthManufact","Java"). The relationship is fairly linear for price shocks in the range 0-75%.



Example 5: Shocks from a file, and collecting solution values in a HAR file

The previous example sent two solution values from each solution to a unique CSV text file, and then used the FindStr command to gather numbers from those CSV files into a single CSV file which was then opened in Excel. That technique is useful but limited -- it becomes inconvenient if you want to extract whole arrays from each solution.

This example uses the same sequence of import shocks, but this time sltoht uses a *header mapping file* to extract several array variables from each solution and store them in a HAR (SOL) file. At the end the program CMBHAR combines all these HAR files into one. The batch file, ImpPrice2.bat, reads:

```
SETLOCAL ENABLEDELAYEDEXPANSION
SET /A COUNT=0
echo off
del sim*.sol
del combined.har
del cmbhar.inp
echo. >>cmbhar.inp
echo 34 >>cmbhar.inp
FOR /F  %%a IN (impprice.shk) DO (
SET /A COUNT+=1
echo at job !COUNT! with shock=%%a
term -cmf impprice.cmf -p1=%%a  >nul
if errorlevel 1 echo Simulation failed
echo running sltoht -map=impprice2.map impprice.sl4 sim!COUNT!.sol
sltoht -map=impprice2.map impprice.sl4 sim!COUNT!.sol >nul
echo sim!COUNT!.sol >>cmbhar.inp
)
echo combined.har >>cmbhar.inp
cmbhar <cmbhar.inp >nul
echo output in file combined.har
```

Type

```
ImpPrice2.bat
```

to run the job. Then examine the files combined.har and sim20.sol using ViewHAR, and files ImpPrice2.map and cmbhar.inp using TABmate.

New points to note above are:

- The 2 statements beginning SET /A COUNT establish a variable COUNT which increments by one each time a shock value is read from ImpPrice.shk. Because the second SET /A COUNT is within the do loop parentheses, we need to start the BAT with the statement SETLOCAL ENABLEDELAYEDEXPANSION and thereafter when count is to be translated (into a number) we need to place it between exclamation marks[3].
- Each simulation creates a file impprice.sl4, from which sltoht extracts results for variables pfimp, ximps and xtot. and places them in a sequence of files sim1.sol, sim2.sol,.and so on.
- cmbhar unites all the sol files into one large file combined.har (each array has an extra index).

The file cmbhar.inp which is input to cmbhar needs to contain the following lines

- a blank line
- the number of sol files (34 in this case)
- the name of each sol file, one per line
- the name of the output file

[3] Don't worry if you can't understand this -- hardly anyone does. Google "stack overflow for setlocal" to find out more.

The commands resembling:

```
echo   . . . . >>cmbhar.inp
```

are used to construct this file within the script.

You could use similar techniques to construct batch files which

- gathered results from a timeseries of sl4 files into one big HAR file
- performed "Monte Carlo" simulations (examining the distribution of results computed using shocks drawn randomly from a known distribution).

Example 6: How results depend on elasticities

Previous examples have run repeated simulations, *varying the shock*. This example uses the same shock each time (100% tax on beer) but varies (from 0 to 40) the elasticity of substitution, SIGMA, between beer and other goods. Theory teaches that the tax will cause a welfare loss, which at first increases in proportion to SIGMA, but then reduces as SIGMA gets large. A small model, modelling a CES consumer, is specified in TAXLOSS.TAB. SIGMA is read from header "SIG" on file SIGMA.HAR. PostSim Write statements put the welfare loss and the SIGMA value to an output HAR file.

The BAT file TAXLOSS.BAT reads:

```
echo off
del out*.har
del combined.har
del cmbhar.inp
echo. >>cmbhar.inp
echo 40 >>cmbhar.inp
for /f "tokens=1,2"   %%a IN (sigma.txt) do (
echo at job %%a with sigma=%%b

echo 1 Real SpreadSheet Header "SIG "; >sig.txt
echo %%b >>sig.txt
txt2har sig.txt sigma.har

taxloss -cmf taxloss.cmf -p1=%%a  >nul
if errorlevel 1 echo Simulation failed
echo out%%a.har >>cmbhar.inp
)
dir/w out*.har
echo combined.har >>cmbhar.inp
cmbhar <cmbhar.inp >nul
echo output in file combined.har
```

Yet another type of FOR command is used here. FOR reads 2 values from each line of SIGMA.TXT -- a counter 1,2,3...40 (which is referred to as %%a) and a sigma value between 0 and 40 (which is referred to as %%b). Within the FOR loop:

- The 3 lines

```
echo 1 Real SpreadSheet Header "SIG "; >sig.txt
echo %%b >>sig.txt
txt2har sig.txt sigma.har
```

write a GEMPACK text file SIG.TXT, with one scalar header SIG, containing the sigma value %%b. Then the program Txt2Har converts SIG.TXT into a HAR file SIGMA.HAR.

- The simulation is run, with the p1 commandline parameter used to name the output file which contains Sigma and welfare values.
- As in the previous example, the command:

```
echo out%%a.har >>cmbhar.inp
```

builds up a list of files for cmbhar to combine into one.

Type

```
TaxLoss.bat
```

to run the example -- then examine the file Combined.har. The following graph was produced: