Performing multiple simulations using CMF parameters with GEMPACK 11.2: the mystery of Horridge's kink

Michael Jerie

Centre of Policy Studies, Monash University

12th June 2013

GTAP 16th Annual Conference on Global Economic Analysis, Shanghai

GEMPACK release 11.2 (May 2013)

- Mainly a maintenance release – includes all bug fixes and improvements to existing features since the previous release 11.1, May 2012.
- Also includes some new features
    - replaceable parameters in CMF files
    - longer TAB lines
    - TABmate Run CMF function
    - closure summary from solution files
    - sparse sorting in reverse order
    - support for Intel Fortran 13
    - and other minor changes …

GEMPACK Release 11.2 allows you to pass replaceable parameters from the command line to CMF files.

CMF syntax: in the CMF file the 10 special parameters

```
<p0>, <p1>, <p2>, … , <p9>
```

Are replaced by values you specify on the command line

Command-line or BATCH syntax: on the command line use optional arguments `-p<digit>="paramter value"` where <digit> is 0,1,3,…,9.

```
mymodel -cmf sim.cmf -p0=par1 -p1=par2 … -p9=par9
```

Quotes around the parameter values are allowed and are *required* if the value contains spaces.

Most examples presented today are available in CoPS archive item tpmh0129.zip

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

The web page

http://monash.edu/policy/gp-bat.htm

contains a basic introduction to BAT file use with GEMPACK.

Example 1: Specifying a shock from the command line

At the command prompt type

```
term –cmf basic1.cmf –p1=10
```

At runtime the shock statement (in basic1.cmf)

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

Is translated to

```
shock pfimp("machines",ORG) = uniform 10;
```

Example 1 continued …

More simulations with different shocks can be run:

```
term -cmf basic1.cmf -p1=20
term -cmf basic1.cmf -p1=30
term -cmf basic1.cmf -p1=40
...
```

Problem: each simulation produces the same output file basic1.sl4.

Example 2: Specifying *shock* and *solution file* from the command line

At the command prompt type

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

This time the simulation produces a solution file sim1-10.sl4.

Command file basic2.cmf contains

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

which at runtime is translated into

```
solution file = sim1-10;
```

Now we can run multiple simulations preserving all solution files.

This can be automated by using a batch file (.BAT file).

Example 2 continued

Example batch file *basic2.bat*:

```
01:  del sim*.sl4
02:  term -cmf basic2.cmf -p2=sim1 -p1=20 >nul
03:  term -cmf basic2.cmf -p2=sim2 -p1=30 >nul
04:  term -cmf basic2.cmf -p2=sim3 -p1=40 >nul
05:  dir sim*.sl4
```

- The job begins by deleting all old solution files!
- The ">nul" redirects output from the terminal to the bin, making it easier to see which simulation is running.

Say we are working in directory C:\tpmh0129, then to run the batch file type basic2

```
C:\tpmh0129\> basic2
```

At the command prompt.

Example 2 continued …

The same task can be achieved by using a loop in the batch file to run the simulations.

Example batch file *basic2a.bat*:

```
01:  rem Example using the batch FOR command
02:  del shk*.sl4
03:  FOR  %%a IN (10,20,30,40,50) DO (
04:  term -cmf basic2.cmf -p1=%%a -p2=shk >nul
05:  )
06:  dir shk*.sl4
07:  rem see variable "NatMacro" in ViewSOL
08:  start viewsol shk*.sl4
```

- The **FOR** command sets variable %%a to 10 then executes the (…) command(s) after D0, then %%a is set to 20 and the DO commands executed again, then 30,40 and 50.
- (obscure) The **START** command is used to open ViewSOL (viewing all solutions) but allows control to return to the command prompt without first closing ViewSOL.

Example 3: Testing different solution methods for speed

This example runs a simulation with different combinations of four Yes|No solution method related CMF options to determine which combination has the best solution time. Number of simulations is 2x2x2x2=16.

Command-line example:

```
term -cmf solmethod.cmf -p1=Y -p2=N -p3=N -p4=N
```

solmethod.bat contains:

```
term -cmf solmethod.cmf -p1=%%a -p2=%%b -p3=%%c -p4=%%d >nul
```

CMF file *solmethod.cmf* contains:

```
IZ1 = <p1> ;  ! Default is YES: ignore zero coefficients at step 1;
                                        NO assists with pivot reuse
LU decompose transpose = <p2> ;       ! default is NO
Markowitz pivots in MA48 = <p3> ;  ! the default is NO
m28 = <p4> ;   ! Default is "NO" which means "use MA48".
log file = SOL_IZ1<p1>_TRN<p2>_MAR<p3>_M28<p4>.log;
```

Running all possible combinations of four binary parameters is best handled with a batch file.

Example 3 continued …

Example batch file *solmethod.bat*:

```
01:  echo off
02:  del *.log
03:  FOR  %%a IN (Y,N) DO (
04:  FOR  %%b IN (Y,N) DO (
05:  FOR  %%c IN (Y,N) DO (
06:  FOR  %%d IN (Y,N) DO (
07:  echo computing with IZ1=%%a  LUtranspose=%%b  MarkowitzPivots=%%c  M28=%%d
08:  term -cmf solmethod.cmf -p1=%%a -p2=%%b -p3=%%c -p4=%%d >nul
09:  if errorlevel 1 echo Invalid combination of options: LUtranspose AND M28
10:  )
11:  )
12:  )
13:  )
14:  echo Solution times were:
15:  findstr /I /C:"Total CPU" sol*.log
16:  rem LU decompose transpose = yes ; only works with MA48 (not with MA28)
17:  rem Markowitz pivots applies only to MA48; does not affect MA28
```

Example 3 continued …

Batch file notes:

- 4 nested **FOR** loops
- Use of the **FINDSTR** command to extract lines from log files

Running solmethod.bat eventually results in:

```
SOL_IZ1N_TRNN_MARN_M28N.log:  [Total CPU is 7.4412475 seconds.]
SOL_IZ1N_TRNN_MARN_M28Y.log:  [Total CPU is 7.7220492 seconds.]
SOL_IZ1N_TRNN_MARY_M28N.log:  [Total CPU is 11.216472 seconds.]
SOL_IZ1N_TRNN_MARY_M28Y.log:  [Total CPU is 7.6284490 seconds.]
SOL_IZ1N_TRNY_MARN_M28N.log:  [Total CPU is 12.698482 seconds.]
SOL_IZ1N_TRNY_MARY_M28N.log:  [Total CPU is 7.1292458 seconds.]
SOL_IZ1Y_TRNN_MARN_M28N.log:  [Total CPU is 7.6128488 seconds.]
SOL_IZ1Y_TRNN_MARN_M28Y.log:  [Total CPU is 24.507757 seconds.]
SOL_IZ1Y_TRNN_MARY_M28N.log:  [Total CPU is 18.657719 seconds.]
SOL_IZ1Y_TRNN_MARY_M28Y.log:  [Total CPU is 24.445356 seconds.]
SOL_IZ1Y_TRNY_MARN_M28N.log:  [Total CPU is 14.242891 seconds.]
SOL_IZ1Y_TRNY_MARY_M28N.log:  [Total CPU is 10.327266 seconds.]
```

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

This example runs multiple simulations, each taking a shock value from a file and then collecting a result so that all results can be used to make an Excel graph.

Batch file contains:

```
term -cmf impprice.cmf -p1=%%a
```

CMF file contains:

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

In this example 34 simulations are run so is best handled by using a batch file.

The batch file will take the shock values from the file *impprice.shk*.

Example 4 continued

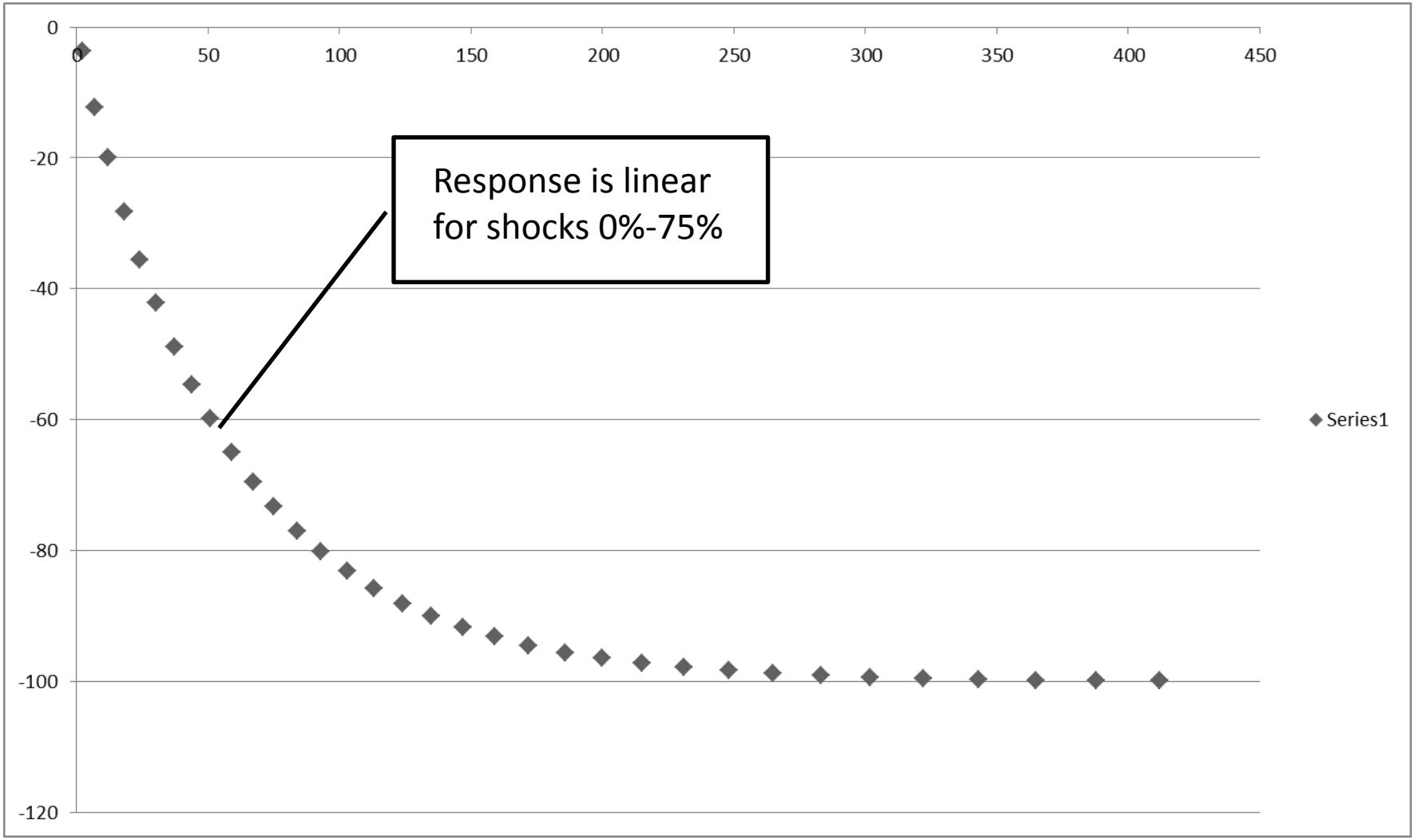
Example batch file *impprice.bat*:

```
01:  echo off
02:  del imp*.csv
03:  FOR /F  %%a IN (impprice.shk) DO (
04:  echo computing with shock=%%a
05:  term -cmf impprice.cmf -p1=%%a  >nul
06:  if errorlevel 1 echo Simulation failed
07:  sltoht -sss -map=impprice.map impprice.sl4 impprice%%a.csv  >nul
08:  )
09:  echo results in  impgraph.csv
10:  findstr /I /C:"java" imp*.csv >impgraph.csv
```

- The **FOR /F** command processes a file. Variable %%a takes values from each line of impprice.shk
- The **SLTOHT** command extracts two values from impprice.sl4 and writes them to a csv file:
  `sltoht -sss -map=impprice.map impprice.sl4 impprice%%a.csv`
- The **FINDSTR** command extracts and collects all results into the file impgraph.csv.

Example 4 continued

ximps (% change quantity) vs pfimp (%-change price)



Example 4a: (Extends example 4) Shocks from a file, and graphing solution values in Excel This example runs multiple simulations, each taking a shock value from a file and varying the number of steps in the solution.

Results are collected to make a chart.

Batch file contains:

```
term -cmf impprice-steps.cmf -p1=%%a -p2=shk -p3=%%i
```

CMF file contains:

```
steps= <p3>;
solution file = shk-<p1>-steps-<p3>;
shock pfimp("OthManufact",ORG) = uniform <p1>;
```

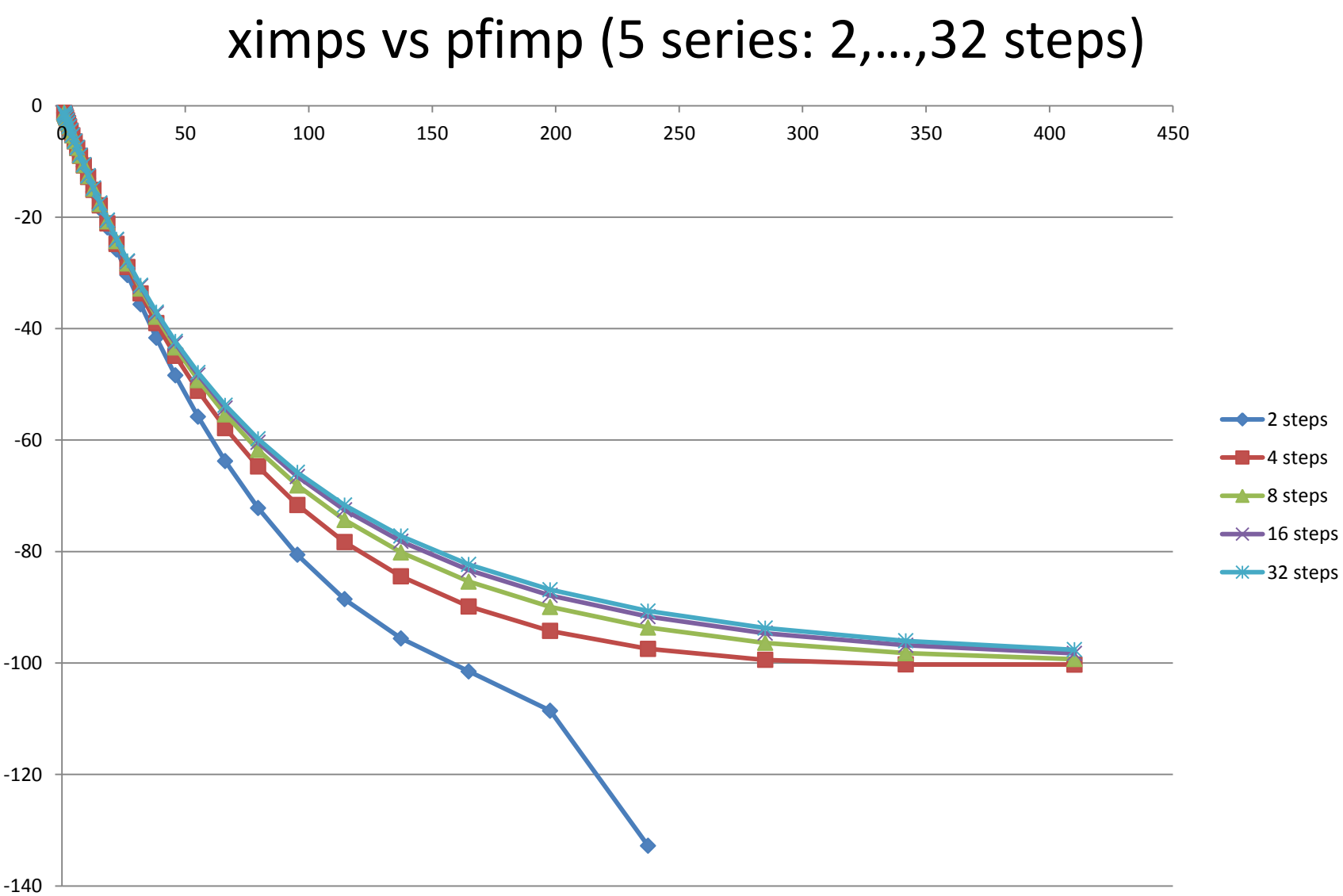
In this example 34 x 5 simulations are run so is best handled by using a batch file.

The batch file will take the shock values from the file *impprice.shk*.

Example 4a continued …

```
01:  echo off
02:  del shk*.csv
03:  FOR %%i in (2,4,8,16,32) DO (
04:  del shk-*-steps-%%i.sl4
05:  del impprice-steps-%%i-*.csv
06:  FOR /F  %%a IN (impprice.shk) DO (
07:  echo computing with steps=%%i shock=%%a
08:  echo on
09:  term -cmf impprice-steps.cmf -p1=%%a -p2=shk -p3=%%i  >nul
10:  if errorlevel 1 echo Simulation failed
11:  echo off
12:  sltoht -sss -map=impprice.map shk-%%a-steps-%%i.sl4 impprice-steps-%%i-%%a.csv
>nul
13:  )
14:  echo results were:
15:  findstr /i "java" impprice-steps-%%i-*.csv
16:  findstr /i "java" impprice-steps-%%i-*.csv >impgraph-steps-%%i.csv
17:  )
18:  del impgraph-steps-all.csv
19:  for %%a in (2,4,8,16,32) do type impgraph-steps-%%a.csv >> impgraph-steps-all.csv
```

Example 4a continued



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

This example runs the same simulations as example 4, this time SLTOHT uses a *header mapping file* to extract several variables from each solution.

Batch file *impprice2.bat* contains:

```
term -cmf impprice.cmf -p1=%%a
```

CMF file *impprice.cmf* contains:

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

Example 5 continued

Example batch file *impprice2.bat*:

```
01:  SETLOCAL ENABLEDELAYEDEXPANSION
02:  SET /A COUNT=0
03:  echo off
04:  del sim*.sol
05:  del combined.har
06:  del cmbhar.inp
07:  echo. >>cmbhar.inp
08:  echo 34 >>cmbhar.inp
09:  FOR /F  %%a IN (impprice.shk) DO (
10:  SET /A COUNT+=1
11:  echo at job !COUNT! with shock=%%a
12:  term -cmf impprice.cmf -p1=%%a  >nul
13:  if errorlevel 1 echo Simulation failed
14:  echo running sltoht -map=impprice2.map impprice.sl4 sim!COUNT!.sol
15:  sltoht -map=impprice2.map impprice.sl4 sim!COUNT!.sol >nul
16:  echo sim!COUNT!.sol >>cmbhar.inp
17:  )
18:  echo combined.har >>cmbhar.inp
19:  cmbhar <cmbhar.inp >nul
20:  echo output in file combined.har
```

Example 5 continued

Batch file notes:

- A loop counter variable COUNT values set using SET /A
- SETLOCAL ENABLEDELAYEDEXPANSION and !COUNT! so that correct values of COUNT are used
- Each simulation creates a solution file *impprice.sl4*
- SLTOHT extracts some results from impprice.sl4 and writes them to files *sim1.sol*, *sim2.sol* …
- After loop is finished CMBHAR combines all sol files into *combined.har,* each header has extra dimension corresponding to sim number
- The CMBHAR input file *cmbhar.inp* is written during execution of the batch file, containing a blank line, number of files to be combined, each file name, and, output file name.

Example 5 continued

Output file *combined.har*

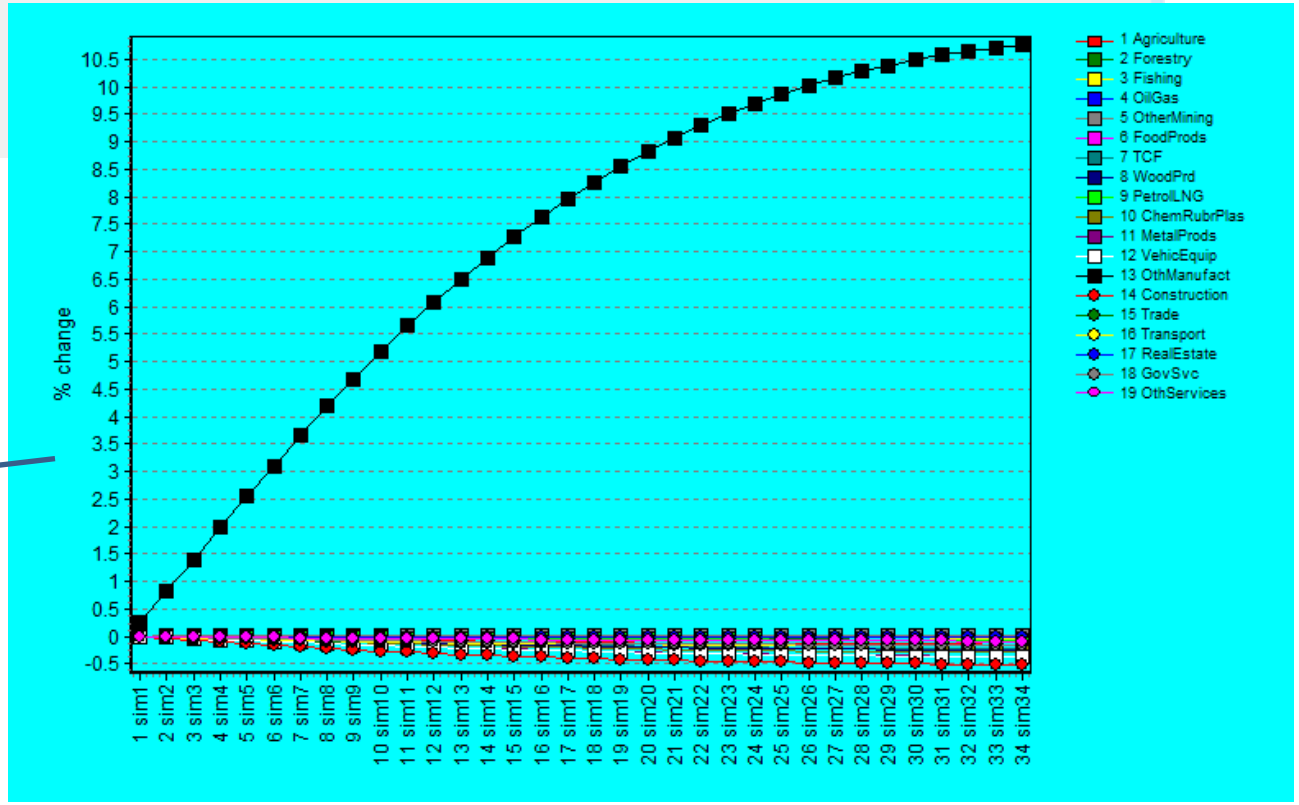
New COMBINED dimension: sim1, sim2, sim3, …

combined.har in C:\gptalk-gp11.2\impprice

File Contents Edit Sets Export Import History Search Aggregation Programs Help

	Header	Type	Dimension	Coeff	Total	Min	Max	Name
1	0001	RE*	COM*ORG*COMBINED	pfimp	31896	0	412	pfimp # Import prices, foreign currency #
2	0002	RE	COM*DST*COMBINED	ximps	-14837	-99.9	7.43	ximps # Volume of imports used in d #
3	0003	RE	IND*DST*COMBINED	xtot	1137	-1.21	14.0	xtot # Industry outputs #
4	CSET	1C	34 length 12					Set COMBINED

Industry output xtot(i) in Java, responding to increasing shock value.



Example 6: Investigating Tariff related Welfare loss as a function of the elasticity parameter.

- Using the GTAP database with all tariffs removed
- A tariff shock
  shock tms = uniform 1;
  uniform 1% world tariff increase is imposed
- Elasticity parameters ESUBD and ESUBM are varied from 0 to "infinity" giving a series of simulation results
- Theory tells us that welfare loss will be an increasing then decreasing function of the elasticities

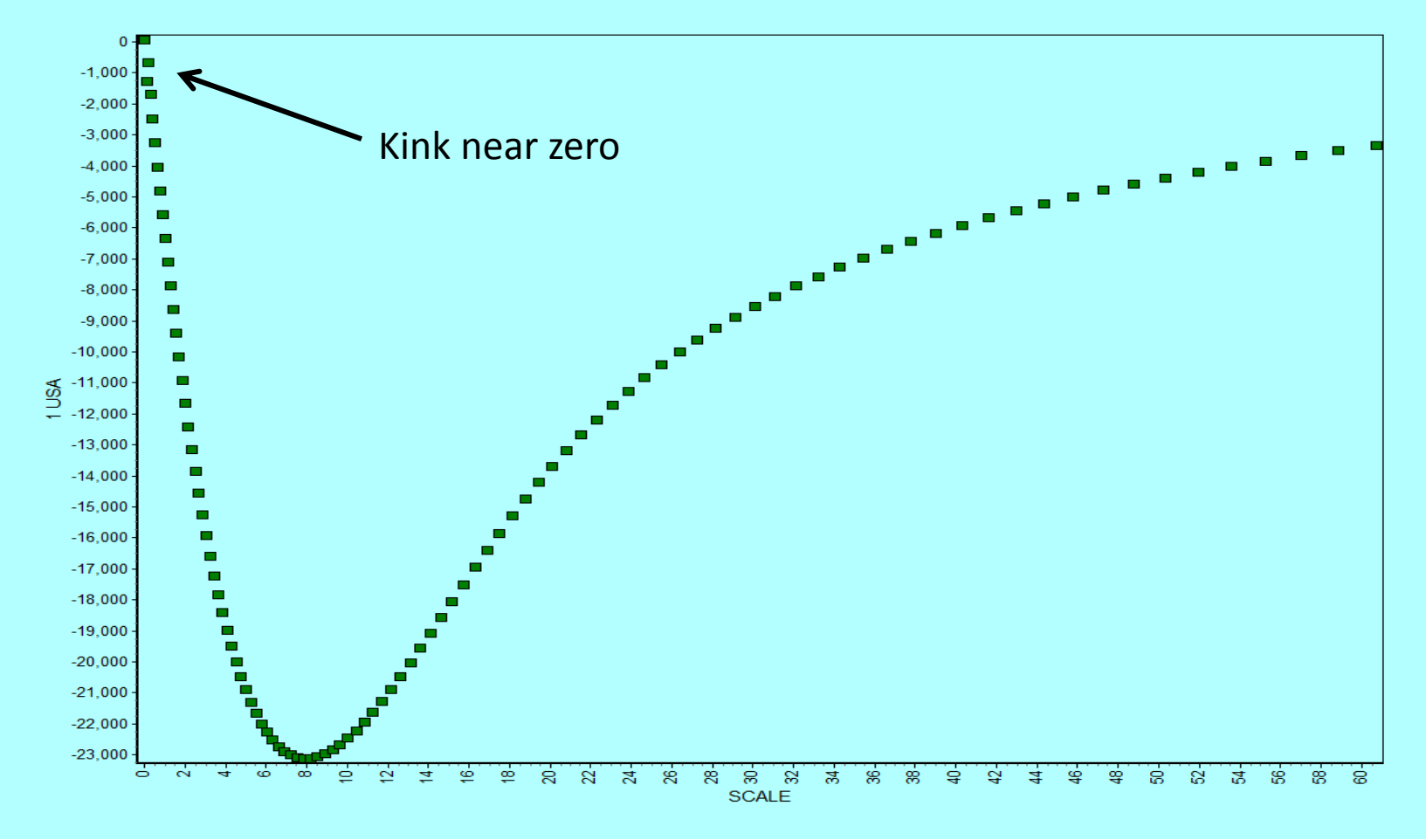
GEMPACK 11.2: Parameter substitution in CMF files

Example 6 continued.

GTAP 6.2 – 3 region basedata.har.

`CNTalleffr`(USA) total contribution to regional EV of allocative effects -- world tariff increase

`CNTalleffr`(USA) vs SCALE factor for elasticities ESUBD ESUBM



GEMPACK 11.2: Parameter substitution in CMF files

Example 6 continued.

`CNTalleffr(USA)` total contribution to regional EV of allocative effects – world tariff increase. Horridge's kink mystery: singularity for SCALE ≈ 0.125.

`CNTalleffr(USA)` vs SCALE factor for elasticities ESUBD ESUBM