Release 10.0 of GEMPACK:

New Features and Changes from Release 9.0

GEMPACK Document No. GPD-9

J.M. Horridge

M. Jerie

K. R. Pearson

Centre of Policy Studies and Impact Project
Monash University, Melbourne, Australia

First edition
May 2008

© Copyright 2008. CoPS/Impact and KPSOFT.

First edition
May 2008

ISSN 1030-2514

ISBN 0-7326-15054

This is part of the documentation of the GEMPACK Software System for solving large economic models, developed at the Centre of Policy Studies/ IMPACT Project, Monash University, Clayton Vic 3800, Australia.

Abstract

GEMPACK is a suite of general-purpose economic modelling software especially suitable for applied general equilibrium models. It can handle a wide range of economic behaviour. It also contains powerful capabilities for solving intertemporal models. GEMPACK provides software for calculating accurate solutions of an economic model, starting from an algebraic representation of the model.

This document describes the new features in Release 10.0 (May 2008) of GEMPACK and the changes from Release 9.0 (April 2005).

The new features included are listed below.

Release 10.0 of GEMPACK contains several enhancements to help you solve large models:

- **Parallel processing.** Most new Windows PCs have two or more processors. If you have such a PC, and if (as is common) you wish to solve a model by extrapolating from 2 or 3 multi-step calculations, you can ask the software to carry out one or two of these multi-step calculations in parallel with the main program. In this way you may be able to solve the model significantly more quickly than normal. Read more about this in chapter 2.
- **Intel Fortran compiler supported.** GEMPACK now allows you to use the Intel Fortran compiler. Indeed, if you are purchasing a new compiler, we recommend it: it seems to produce faster-running EXEs (executable files). In addition, it allows you to produce either 32-bit or 64-bit EXEs (see next point). Read more about Intel Fortran in chapter 3.
- **64-bit TABLO-generated programs.** Previous versions of GEMPACK for Windows PCs have supported the Lahey compiler, which produces 32-bit EXEs. A 32-bit program cannot use more than 2GB of memory – placing an upper limit on the size of your model. GEMPACK now allows you to use a 64-bit EXE, as long as (a) you have the Intel Fortran compiler, and (b) you are running a 64-bit version of Windows. Read more in chapters 3 and 4.
- **Better re-use of pivots.** For some time GEMPACK has supported pivot re-use during the LU phase of solution. The new approach allows pivots to be re-used more frequently, saving time in multistep simulations. Read more in chapter 5.
- **Sparse format for Header Array files.** This option can in some cases greatly reduce the size of HAR files. Read more in chapter 6.

Other minor enhancements are described in chapters 7 and 8, including:

- support for Windows Vista (section 8.1)
- more detailed reporting of arithmetic errors (section 7.4)
- a Condensation Information File which could help you choose which variables to substitute or omit (section 7.8)
- new functions for the Log-Normal Distribution (section 7.9) and to RAS a matrix (section 7.10)
- more consistent treatment of integers in TAB files – a few previously OK TAB files may need simple changes as a consequence (section 7.11)
- Command file required if TAB file contains equations (section 7.15)
- you can speed up Complementarity sims if you use a single Euler calculation (section 7.18)
- AnalyseGE loads a linearised TAB file if your TAB file contains levels equations (section 8.2.4)
- RunDynam/RunMONASH can simultaneously run 3 jobs (section 8.2.5)

Document Attributes

Name	Release 10.0 of GEMPACK: New Features and Changes from Release 9.0
Audience	CGE Modellers
Identifier	GPD-9

History :

Date	Author(s)	Comment
May 2008	M.Horridge, M.Jerie & K.Pearson	First Edition (GPD-9) [Release 10.0]

Table of Contents

CHAPTER 1

1 Release 10 of GEMPACK – Introduction and Overview

This document describes the new features in Release 10.0 (May 2008) of GEMPACK and the changes from Release 9.0 (April 2005).

Changes between Release 9.0 and Release 8.0 of GEMPACK (October 2002) are described in the companion document GPD-5 (April 2005).

All references to the GEMPACK documents GPD-1, GPD-2, GPD-3, GPD-4 and GPD-8 refer to the Release 8.0 versions dated October 2002. References to GPD-5 refer to the Release 9.0 versions dated April 2005. The only new documents for Release 10 are this document GPD-9 and the installation documents GPD-6 and GPD-7.

1.1 Contacting the Developers

For more information about GEMPACK, see the web page at:

http://www.monash.edu.au/policy/gpcont.htm

or write directly to:

Professor Mark Horridge
Centre of Policy Studies and Impact Project
Monash University
Clayton, Vic 3800
Australia
email: Mark.Horridge@buseco.monash.edu.au

1.2 Different Versions of GEMPACK and Associated Licences

The versions and licences for Release 10.0 are the same as described in section 1.9 of GPD-1, except that the "demonstration version" has been discontinued – instead the Limited Executable-Image version is now freely downloadable and comes with a temporary licence lasting for 3 months.

However new licence files are required for use with Release 10.0 of GEMPACK.

1.3 Guide to the Documentation

We provide two alternative guides. Readers new to GEMPACK should read section 1.3.1 below, while readers familiar with Release 9.0 (or a previous version) of GEMPACK should read section 1.3.2.

This document contains **an index** which should make it easier for you to find information. The other GEMPACK documents also contain detailed indexes. The index in each document contains pointers into the document itself, and also into other GEMPACK documents.

1.3.1 For New GEMPACK Users – Getting Started

Start with document GPD-1 if you are a new GEMPACK user. Follow the guide in section 1.5.1 of GPD-1.

You should not begin reading this document GPD-9 until after you have worked through the parts of chapters 2, 3 and 4 of GPD-1 recommended in section 1.5.1 of GPD-1.

This document GPD-9 assumes that you are familiar with GEMPACK and that you have worked with it for some time.

1.3.2 Guide For Experienced GEMPACK Users

If you have worked with Release 9.0 (or an earlier version) of GEMPACK, the main thing you will want to see is a list of the new features. The new features in Release 10.0 are listed in section 1.4 below. If you have not used Release 9.0, you will find a list of the new features introduced in Release 9.0 in section 1.4 of GPD-5 and in the previous releases in sections 1.6 and 7.1-7.4 of GPD-1.

A few GEMPACK Source-code users may be interested in writing their own Fortran programs using GEMPACK subroutines. If you are so interested, please see Harrison *et al* (2005).

1.4 New Features in Release 10.0

Release 10.0 of GEMPACK contains several enhancements to help you solve large models:

- **Parallel processing.** Most new Windows PCs have two or more processors. If you have such a PC, and if (as is common) you wish to solve a model by extrapolating from 2 or 3 multi-step calculations, you can ask the software to carry out one or two of these multi-step calculations in parallel with the main program. In this way you may be able to solve the model significantly more quickly than normal. Read more about this in chapter 2.
- **Intel Fortran compiler supported.** GEMPACK now allows you to use the Intel Fortran compiler. Indeed, if you are purchasing a new compiler, we recommend it: it seems to produce faster-running EXE files. In addition, it allows you to produce either 32-bit or 64-bit EXEs (see next point). Read more about Intel Fortran in chapter 3.
- **64-bit TABLO-generated programs.** Previous versions of GEMPACK for Windows PCs have supported the Lahey compiler, which produces 32-bit EXEs. A 32-bit program cannot use more than 2GB of memory – placing an upper limit on the size of your model. GEMPACK now allows you to use a 64-bit EXE, as long as (a) you have the Intel Fortran compiler, and (b) you are running a 64-bit version of Windows. Read more in chapter 4.
- **Better re-use of pivots.** For some time GEMPACK has supported pivot re-use during the LU phase of solution. The new approach allows pivots to be re-used more frequently, saving time in multistep simulations. Read more in chapter 5.
- **Sparse format for Header Array files.** This option can in some cases greatly reduce the size of HAR files. Read more in chapter 6.

Other minor enhancements are listed in chapters 7 and 8, including:

- support for Windows Vista (section 8.1)
- more detailed reporting of arithmetic errors (section 7.4)
- a Condensation Information File which could help you choose which variables to substitute or omit (section 7.8)
- new functions for the Log-Normal Distribution (section 7.9) and to RAS a matrix (section 7.10)
- more consistent treatment of integers in TAB files – a few previously OK TAB files may need simple changes as a consequence (section 7.11)
- Command file required if TAB file contains equations (section 7.15)
- you can speed up Complementarity sims if you use a single Euler calculation (section 7.18)
- AnalyseGE loads a linearised TAB file if your TAB file contains levels equations (section 8.2.4)
- RunDynam/RunMONASH can simultaneously run 3 jobs (section 8.2.5)

1.5 The GEMPACK Programs

Release 10 of GEMPACK contains the same programs as are listed in section 1.7 of GPD-1.They fall into two broad groups:

1.5.1 The Original Command-line Programs

The original GEMPACK programs, all written in Fortran, are command-line programs (without a graphical interface). They form the core of GEMPACK and are still directly used by experienced modellers. They are portable between different operating systems. The chief programs are:

Program	Description
TABLO	translates a TAB file into an executable program (or, optionally, into bytecode [GEMSIM Auxiliary files]).
GEMSIM	an interpreter: executes bytecode versions of TABLO-generated programs.
MODHAR	translates text data into a HAR (header array) file.
SEEHAR	translates a HAR file into a text file.
SLTOHT	translates an SL4 (Solution) file into a HAR or a text file.

1.5.2 The Windows Programs

Since 1995, additional GEMPACK programs have been available, which only run on Windows PCs. These have a more modern graphical interface, and are written in Pascal [Delphi]. Most of these Windows programs make some use of the core command-line programs listed above: they are really "wrappers" or "shells" which provide a more convenient interface to the original GEMPACK programs. The main Windows programs are:

Program	Description	Using core programs:
WinGEM	a portal to all the other GEMPACK programs, which guides the user through the stages of specifying a model, creating data files, and running simulations.	all
RunGEM	a convenient interface for running simulations and viewing results.	SLTOHT and others
TABmate	a text editor tailored to GEMPACK use.	TABLO
AnalyseGE	used to analyse simulation results: presents an integrated view of input data, model equations, and results.	TABLO, SLTOHT and others
ViewHAR	used to view and modify HAR data files.	none
ViewSOL	used to view simulation results.	SLTOHT (sometimes)

Another GEMPACK Windows program, RunDynam, is used to organize, perform, and interpret multi-period simulations with recursive-dynamic CGE models. RunDynam is not included in the standard GEMPACK package -- it may be purchased separately.

1.6 Models Supplied with GEMPACK

All the models described in section 1.8 of GPD-1 and section 1.6 of GPD-5 are supplied with Release 10.0 of GEMPACK. In addition, some extra ones are supplied, as follows.

1.6.1 New Example Models

GTAP-OQ.ZIP contains a version of GTAP which models output (production) quotas. It includes example simulations with the RunGTAP ACORS3X3 data with output quota data added.

A small number of new TAB and Command files showing examples of arithmetic errors referred to in section 7.4 have been added.

Also supplied with the Examples files is **G61-GP.ZIP**. This file is referred to in section 2.7 of GPD-8 but was omitted from the Examples files supplied with Release 8 and Release 9. We apologise for that omission.

1.7 GEMPACK Web Site

The GEMPACK Web site at:

http://www.monash.edu.au/policy/gempack.htm

contains up-to-date information about GEMPACK, including information about different versions, prices, updates, courses and bug fixes. We encourage GEMPACK users to visit this site regularly.

In particular, this site contains a list of **Frequently Asked Questions (FAQs)** and answers at address

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

This is updated regularly. It is a supplement to the GEMPACK documentation. If you are having problems, you may find the solution there. We welcome suggestions for topics to include there.

There are also alternative GEMPACK web sites at addresses

http://www.gempack.com *http://www.gempack.com.au*

and you can send email to *info@gempack.com* or *support@gempack.com*

At present these alternative web sites merely point to the main GEMPACK site whose address is above, and email to these two email addresses is sent on to Mark Horridge.

1.8 GEMPACK-L Mailing List

GEMPACK-L is a mailing list designed to let GEMPACK users communicate amongst themselves, sharing information, tips etc. The GEMPACK developers occasionally make announcements on it (bugs, new releases etc). The list is moderated to prevent spam.

We encourage all GEMPACK users to subscribe to it. Once you have subscribed, you can send mail messages to all others on the list, and you will receive as mail any messages sent to the list. For information about subscribing and sending messages, please see:

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

1.9 Machines Supported

Most GEMPACK users work on Windows PCs. GEMPACK documents GPD-6 and GPD-7 contain instructions for installing and using the Source-Code and Executable-Image Versions on Windows PCs.

We also supply Source-code GEMPACK for Unix machines. This includes modern Macintosh PCs which run under Apple's Unix-based OS X operating system. If you wish to install GEMPACK on a Unix machine or on a Macintosh PC, please contact GEMPACK support for further details.

1.10 Beta Testers

We are grateful to Michael Bourne, Matt Clark, Robert Ewing, Lindsay Fairhead, Alan Fox, Joseph Francois, Owen Gabbitas, Kevin Hanslow, Jorge Hernandez, Thomas Hertel, .Lars-Bo Jacobsen, Guy Jakeman, Peter Johnson, Terry Maidment, Robert McDougall, Athoula Naranpanawa, Hom Pant, Federica Santuccio, Agapi Somwaru, Marinos Tsigas, Greg Watts, Ian Webb, Alex Whitmarsh Ashley Winston, several of our colleagues at the Centre of Policy Studies, and any others who have been beta testers for Release 10.

CHAPTER 2

2 Solving a Model in Parallel on a Machine with Two or More Processors

Many Windows PCs now have two or more processors. These PCs cost little more than a PC with a single processor. This section only applies to those working on such a machine.

Although for some years Windows PCs have been available which incorporated 2 or more CPUs (each in their own motherboard socket), such PCs were usually large, noisy and expensive, with a rather limited market (server applications). Not much software was adapted to use multiple CPUs.

Recently, dual-core or even quad-core processors which incorporate several processing units in a single package have become popular. First AMD and then Intel has introduced a number of models – used in most new desktop or laptop PCs.

One reason for this trend is that traditional single-core CPUs have reached a performance plateau (or at least improve more slowly than before). Multi-processing offers the prospect of continued performance increases – if software can take advantage of it.

Since GEMPACK Release 9.0, RunDynam/RunMONASH (see section 8.2.5) can run 2 or 3 jobs concurrently, if you have several processors on your PC.

Now GEMPACK Release 10.0 allows you to divide a single simulation between processors.

If you wish to solve a model by extrapolating from 2 or 3 multi-step calculations, you can ask the software to carry out one or two of these multi-step calculations in parallel with the main program. In this way you may be able to solve the model significantly more quickly than normal, as the example below shows.

The separate multi-step calculations are independent of each other. For example, if you are doing Gragg 2,4,6-step calculations, the 6-step calculation can be done independently of the 4-step calculation.

We are grateful to Hom Pant who pointed out that these calculations are independent of each other and who encouraged us to implement parallel calculations to take advantage of this. We are also grateful to ABARE (the Australian Bureau of Agricultural and Resource Economics) for financial support of this project.

Example 1 : Gragg 2,4,6-step. Reducing time by 40-50%

Suppose you are solving using Gragg 2,4,6-step calculations. If you have two processors, you can ask the main program to carry out the 2-step and 4-step calculations. And you can ask the main program to run (on the other processor) a separate job which carries out the 6-step calculation. When that is finished, the main program will read the results from the 6-step calculation and then combine these results with the results of the 2-step and 4-step calculations to do the extrapolation and produce the usual results.

Since the 6-step calculation probably only takes as long as the 2-step followed by the 4-step, this will approximately halve the total elapsed time required to solve the model.

When you do this, we call the program you start running **the master** and we call the jobs it starts (the 6-step calculation in the example above) **the servants**.

Example 2 : Replacing Euler 4-step by more accurate Euler 2,3,4-step in same time

Suppose that you normally solve your large model using only a single Euler 4-step calculation. [This is often done with MONASH-style models.]

If you have two processors, you may now be able to carry out the more accurate Euler 2,3,4-step calculation (more accurate because of extrapolation) in about the same elapsed time as it takes to do the single Euler 4-step calculation. The master program carries out the 2-step and 3-step calculations while the servant program carries out the 4-step calculation.[1]

2.1 You Need Plenty of Memory

When you run two or three programs in parallel (one master and one or two servants), each program requires about the same amount of memory. If your model takes about 500MB of memory to solve normally, you will need about 1000MB to run one servant or about 1500MB to run two servants. If you don't have this much memory, the servants and/or the master will be running in virtual memory (when the hard disk emulates RAM), which will make them run very slowly. Good performance requires that each running task has ample access to RAM.

However, 32-bit CPUs and 32-bit Windows can only manage 4GB of RAM. This limits the possibility of running several memory-intensive tasks simultaneously. 64-bit CPUs and Windows versions are now available which can manage much more memory. Indeed, nearly all the dual-core or quad-core CPUs purchased now are 64-bit capable – *if used in conjunction with 64-bit Windows*. Hence, there is a connection between the transition to multicore PCs and the transition to 64-bit computing. That connection is explored in chapter 4.

2.2 Telling the Program to Run in Parallel

You can do this by including the following statement in your Command file.

```
servants = NO|1|2 ;          ! NO is the default
```

- "**servants = 1 ;**" tells the program to use one servant. The servant will do the longest multi-step calculation.
- "**servants = 2 ;**" tells the program to use two servants to do the longer two multi-step calculations. This only makes sense if (a) you are extrapolating from three (not two) multi-step calculations, and (b) you have three or more processors on your machine (the main program will use one processor and each of the servants ideally has its own processor).
- "**servants = no ;**" tells the program not to use servants. That may be appropriate even if you have two processors because you want to do serious word-processing while the simulation is running and you want the second processor to concentrate on the word processing.

2.2.1 Problems with the Program Deciding How Many Servants

It is not easy for the program to reliably discover how many processors are available on your PC.

Although the program could find out how much memory is available on your PC, it is hard to know early on how much memory will be required to solve your model. Accordingly, the program cannot sensibly decide whether you have sufficient memory for one or more servants.

---

[1] The master does most of the first step (including the LU decomposition) before starting the servant. The servant needs to do only a small part of the first step and then all of steps 2, 3 and 4. While that is happening, the master completes the first step of the Euler 2-step (this is very quick), then does all of the second step of the 2-step. Then the master completes the first step of the Euler 3-step (again very quick), and then does all of the second and third steps of the Euler 3-step. Then the master waits for the servant to finish and completes the run by doing the extrapolation.

Thus the master does a total of about 4 steps (both steps of Euler 2-step then steps two and three of Euler 3-step). Starting after the master has done most of the first step, the servant only has to do 3 full steps (the second, third and fourth steps of the Euler 4-step). Thus the servant is likely to complete the 4-step at about the same time as the master completes the 3-step. You can see that the total elapsed time for the master is about the same as it takes to do 4 steps, so Euler 2,3,4-step with one servant probably takes about as long as a single Euler 4-step.

That is why, at least for the present, we require you to tell the program whether or not to use servants and, if so, how many servants to use.

2.3 Restrictions

- You can only run in parallel if you are carrying out a simulation in which you are extrapolating from two or three separate multi-step calculations. You cannot run in parallel if you are carrying out a simulation using a single multi-step calculation or if you are running a data-manipulation TAB file.

  If you put "servants = 1 ;" or "servants = 2 ;" in a Command file which does not carry out a simulation or which carries out a single multi-step calculation, that statement will be ignored (as a note in the LOG file will indicate).[2]
- You cannot run in parallel if you are using the memory sharing features described in section 4.2 of GPD-5. Running in parallel is only sensible if you have plenty of memory – see section 2.1.
- You cannot run in parallel if you are starting from old Equations and SLC files – see section 7.22.
- If there are complementarity statements in your model, parallelization can only help you on the accurate run, and then only if the accurate run does not include subtotals. You should include subtotal commands on the approximate run (see section 2.6.2 for more about this).

2.4 Servants Work in Their Own Directories

Servant programs always work in a subdirectory of the directory in which the master program is running. That is, servant programs produce most of their output files (and their temporary files) in a subdirectory.[3]

The name of the subdirectory depends on the name of the Solution file being produced by the master program and the number of the multi-step calculation the servant is doing.

Example

> Suppose that the master program is running in directory **`c:\mymodel\32sector`** and that it is producing Solution file **`thissol.sl4`** in that directory. Then, if a servant is doing multi-step calculation number 3 (the last one), it will work in the subdirectory **`thissol-ms3`** of `c:\mymodel\32sector`.

The main two outputs from the servant are

- the results file which is called **`servant-output.sl4`** and which is produced in the subdirectory in which the servant is working. [Although this file has suffix .sl4, it is not a true Solution file. For example, you cannot open it with ViewSOL. It only contains the numerical results (cumulative solution and any subtotals solutions) for one multi-step calculation. It does not contain the other information (for example, details of closure and shocks) usually contained in a Solution file.]
- the relevant updated data files, which are produced where the master would produce them if it was running normally, not as a master. These have suffixes **.ud6** (for the second multi-step calculation) or **.ud7** (for the third multi-step calculation). These do not go in the subdirectory in

[2] That means it is safe to put "servants = xx ;" in CMFSTART files under RunDynam, RunMONASH, RunGTAP (see section 2.10). Such a statement will not cause an error when the software runs the model program in order to set up closures etc.

[3] Technically, the working directory of the servant is the same as that of the master. But the servant produces most of its output files (and all of its temporary files) in the subdirectory.

which the servant is working but in the same directory as where the master will produce the final updated file.

When the servant program finishes its run, the master program reads the results and then (normally) deletes the subdirectory in which the servant was working.

2.5 Master Log File is Complete

If you ask for a LOG file, the log file produced by the master normally includes a complete log file for each of the servants. These begin with a heading of the form

```
&&& SERVANT LOG &&
```

However:

- If there is a fatal error (see section 2.7), the master log file may not contain the log from each of the servants.
- If you are doing automatic accuracy, the master log file for each subinterval normally contains the log file from one of the servants. But if the master finds a Coefficient out of range, the master may not wait until the servants have finished before redoing the subinterval, so the log file for that subinterval may not include details from the servants.

2.6 When the Servants Start and Finish

The master program carries out all of the first pass up to the point where it has solved the equations for all multi-step calculations. [The LHS is the same for each different multi-step calculation. The master calculates the two or three different right-hand sides and solves them all.]

Then it starts the servants running (and it keeps going on the first multi-step calculation).

Each servant stops when it has solved all steps relevant to its own multi-step calculation.

When the master has completed the separate multi-step calculations it is responsible for (that may be one or two multi-step calculations), it waits until the servants have all finished. Then it reads the results from the servants and continues to do the extrapolation and any post-simulation processing required.

2.6.1 If There are Several Subintervals

The procedure in section 2.6 above is repeated for each subinterval. The servants carry out the different steps of one multi-step calculation for a single subinterval and then stop. Then they are called into action again to do the same multi-step calculation in the next subinterval, and so on.

2.6.2 If There are Complementarity Statements

The approximate calculation (in each subinterval) is a single multi-step Euler calculation. The master must do this (since there is nothing for the servants to do). Only the accurate calculation can use servants to do two or three multi-step calculations.

This means that the time saving is not as great when there are Complementarity statements in the model, since there is only a time saving on the accurate run, not on the approximate run.

2.6.3 If Using Automatic Accuracy

This is much the same as several subintervals (see section 2.6.1). The servants are called on each subinterval.

The complication (it is a considerable one) is that, if the value of a Coefficient goes out of range (see sections 6.4 and 7.4 of GPD-3) in one of the multi-step calculations, all the others must stop and the subinterval must be done again (with a shorter length). This requires good communication between the master and the servant.

For example, if a servant detects a Coefficient out of range, it must tell the master to stop and redo the subinterval. And the master must tell the other servant (if there is one) to stop.

The other criterion used in automatic accuracy simulations is the accuracy – see section 7.4 of GPD-3. The master is the one which does the extrapolation so it does not need to communicate with the servants when deciding whether or not to redo a subinterval because of insufficient accuracy. [This can only be decided after the extrapolation, when no servants are running.]

2.7 If a Fatal Error Occurs

If one of the servants encounters a fatal error, it must stop and tell the master which must also stop. The master may not be able to immediately stop any other servants which are running. The master will give the servant instructions to stop. However the servant only checks this one or two times during each step of its multi-step calculation.

If the master encounters a fatal error, it must tell the servants to stop and then stop itself. Again the master will instruct each servant to stop – but again the instructions may not be obeyed immediately.

2.8 Fine Print

2.8.1 Displays and Writes to Terminal At All Steps

This relates to option **DWS** or Command file statement "**dws = yes ;**" – see section 6.1.9 of GPD-3.

A master program will do as requested. But any servant programs it runs will not do any Displays. However servant programs will do writes to the terminal for all steps that they carry out (as will the master program).

2.8.2 Multi-Step Calculations May Not Be Independent of Each Other

With the new (Release 10) re-use of pivots strategy (see chapter 5), the separate multi-step calculations are not completely independent of each other – see the first footnote to point number 2 in section 5.1.

2.9 Some Elapsed Times

In this section we report some elapsed times with and without servants for various models with different compilers.

All times reported here were obtained on a Windows XP PC with two dual-core AMD64 chips[4] (that is, 4 processors) with 8GB of physical memory.

The times for the LF90, LF95 and Intel-32 compilers were obtained running under the Windows XP Professional (32-bit) operating system. The times for the Intel-64 compiler were obtained running under the Windows XP Professional x64 Edition (64-bit) operating system.

In each case, the default compiler options as supplied with Release 9.0 of GEMPACK were used. For LF90 and LF95 this is basically O1 optimisation options (the same as for Release 9.0 of GEMPACK

[4] Two Dual Core AMD Opteron Processors 270 2.01GHz.

with these compilers). For the Intel compilers, this was /O2 optimisation option (which is also the default recommended by Intel)[5].

The times reported are elapsed times. Even for the same compiler and Command file, the elapsed time varies from run to run.[6] The times reported are the average of 3 runs in each case.

- The version of **GTAP** is one with 38 regions and 39 tradeable commodities. This is the same 38x39 aggregation we reported about in sections 4.2.9, 5.2.1 and 5.3.1 of the Release 9 version of GPD-5. The simulation in **SIM1.CMF** is a Gragg 2,4,6-step simulation.
- The version of **GTEM** used for the report below has 23 regions and 29 tradeable commodities. [This is the same version of GTEM we reported about in section 5.3.1 of the Release 9 version of GPD-5.] **GTEMSIM3.CMF** is an Euler 3,5,7-step simulation. We are grateful to Guy Jakeman for supplying us with this version of ABARE's GTEM model.
- The **TERM-WATER** version is a version of TERM with 48 commodities and 20 regions aimed at water-related applications. This is the TERM model reported in the tables in sections 5.2.1 and 5.3 of the Release 9 version of GPD-5. This model contains Complementarity statements. **TERMSIM1.CMF** does 3-step Euler approximate run followed by Euler 2,3,4-step accurate run. There are no Complementarity state changes in this simulation.

2.9.1 Comments on the Elapsed Times Reported in Table 2.9

The best you can hope for is that, by using 2 servants, the elapsed time will be the same as the elapsed time for the longest of the 3 separate calculations. As you can see from the last two columns in Table 2.9, that pretty much happens (except for the LF90 GTAP simulation).

With one servant, the master does the shorter two multi-step calculations while the servant does the longest multi-step calculation.

Gragg 2,4,6 Example

> In the case of the Gragg 2,4,6 GTAP simulation, this means that the master does something like 2+4=6 passes while the servant does 6 passes. [A Gragg 4-step has 5 passes. The first pass of all multi-step calculations is done by the master before the servant starts. So the servant does the last 6 passes of the 7-pass Gragg 6-step while, starting at the same time, the master does the last 2 passes of the Gragg 2-step and then the last 4 passes of the Gragg 4-step.] On the basis of this (crude) analysis, you might expect that the master would finish its two tasks (Gragg 2 and 4) at about the same time as the servant finished its one task (Gragg 6). In fact, in this simulation, the master takes longer than the servant because of variations between the times taken for the different steps.[7]

The TERM-WATER simulation has a 3-step Euler approximate run followed by an Euler 2,3,4-step accurate run. The master does all of the approximate run (the servants are not able to help with that). Hence the overall time saving is not as great as for the other simulations reported.

[5] Since that testing, we have reverted to /O1 optimisation as the default with GEMPACK for the Intel compiler – see "compiler options" in chapter 7 of GPD-6.

[6] For example, with Intel 32, the 3 elapsed times for the 2-servant GTAP simulation were 24 m 48s, 21m 58s and 23m 40s, while for Intel 64, the 3 elapsed times for the same 2-servant simulation were 21m 31s, 21m 30s and 21m 36s.

[7] For the Intel-32 compiler, the single step times taken through this simulation vary from about 2m 35s to about 4m 56s. In this case, it just happens that more of the longer ones occur in the parts done by the master. For another simulation, the reverse might happen.

Simulation and Compiler	**Simulation (no servants)**	**1 Servant**	**2 Servants**	**Longest multi-step calculation**
GTAP38X39 SIM1.CMF	**Gragg 2,4,6**			**Gragg 6**
LF90	48m 35s	32m 18s	26m 39s	20m 57s
LF95	66m 24s	35m 19s	34m 58s	33m 58s
Intel 32	48m 10s	26m 28s	23m 49s	23m 40s
Intel 64	41m 23s	24m 55s	21m 32s	20m 48s
GTEM GTEMSIM3.CMF	**Euler 3,5,7**			**Euler 7**
LF90	17m 31s	10m 51s	10m 21s	10m 14s
LF95	46m 8s	25m 32s	24m 57s	24m 19s
Intel 32	19m 43s	11m 3s	10m 57s	10m 45s
Intel 64	18m 20s	10m 39s	10m 38s	9m 54s
TERM-WATER TERMSIM1.CMF	**Euler 2,3,4 (+ 3-step approx run)**			**Euler 4 (+ 3-step approx run)**
LF90	6m 33s	4m 38s	4 m 32s	4 m 33s
LF95	7m 34s	5m 28s	5m 32s	5m 15s
Intel 32	5m 23s	3m 38s	3m 27s	3m 37s
Intel 64	3m 58s	2m 50s	2m 50s	2m 45s

Table 2.9: Elapsed Times for Typical Simulations With and Without Servants

Summary

- If you have 3 or more processors and run with 2 servants, you can expect that the elapsed time will be roughly that for the longest multi-step calculation.
- If you have only 2 processors (or run with only one servant), you still can expect to make considerable time savings. And you can estimate how much by considering how many total passes the master and servant must each make (once the servant is started).

Fine Print

The times reported in Table 2.9 were obtained before the new re-use of pivots strategy (see chapter 5) was introduced.

2.10 Using Master/Servant under RunDynam/RunMONASH etc

You can instruct RunDynam etc to use a master and one or two servants for each solve. To do so, include the relevant statement

`servants = x ;` ! x is 1 or 2

in your CMFSTART file(s).

Your PC will need more memory to use servants, since the master and each servant need the same memory as a single run of the model (see section 2.1). So if you want to run efficiently with one servant, you need to have available on your computer at least twice the memory necessary for one solve of the model. [Otherwise the 2 programs will be fighting each other for memory and the master plus servant may end up taking longer than if you ran without a servant.]

RunDynam etc can now run up to 3 jobs concurrently (see section 8.2.5). Each job requires its own memory. So if you wish to run 3 concurrent jobs efficiently, you need to have available on your computer at least 3 times memory necessary for one solve of the model. [Otherwise the 3 programs will be fighting each other for memory and the 3 jobs concurrently may end up taking longer than if you ran them one after the other.]

If you want to combine concurrent solves and master/servant under RunDynam, you have to be even more aware of the memory requirements.

Example:

> Suppose, for example, you click on the **Run All** button of RunDynam and that you are allowing 3 concurrent solves (under the RunDynam Options menu) and have specified each simulation to use a master and one servant. Then for efficient running you need to have available on your computer at least 6 times the memory necessary for one solve of the model.

CHAPTER 3

3 Intel Fortran Compiler (32/64-bit) Now Supported

GEMPACK now supports the Intel Fortran compiler, as well as the two Lahey compilers LF90 and LF95 previously supported).

The Intel compiler is of particular interest on 64-bit PCs since it comes with a 64-bit version which creates EXE files that will only run on a 64-bit processor with 64-bit Windows. These EXEs can access more than the 2GB limit which applies on 32-bit operating systems. Modellers who have large models which are currently close to the 2GB limit will be free of this limit if they use GEMPACK with the Intel compiler on a 64-bit PC.

The Intel compiler also comes with a 32-bit version.

We recommend (see section 7.1 for more details) that users with LF90 consider upgrading to the Intel compiler.

3.1 Intel Fortran Produces Same Binary Files as LF95

The Fortran standard allows compiler vendors to design their own binary file format. Happily, Intel has opted for the same format as used by LF95[8], so that standard GEMPACK programs and TABLO-generated programs compiled with either Intel or LF95 compilers can share HAR files[9] without any need for format conversion. Each compiler has an X-86 Linux version—which also uses the same binary format.

3.2 Should I Consider Moving to 64-bit Intel Now?

Purchasers of the Intel Fortran compiler actually receive both 32-bit and 64-bit versions. But, to use the 64-bit compiler, you need to be running 64-bit Windows. And 64-bit Windows has some advantages even if you stick with the 32-bit compilers. These issues are explored in the next chapter.

[8] Earlier Lahey compilers, such as LF90, used a different binary format. See chapter 15 of GPD-4.

[9] Standard GEMPACK compile options for Intel include the /fpscomp:logicals flag, which is needed to ensure that Intel stores logical (true/false) values in the same way as LF95. With this compiler option we believe that Header Array files and other binary files (including AXS and GSS files) written by GEMPACK programs compiled with the Intel compiler will be identical to those written by LF95 programs.

CHAPTER 4

4 64-Bit Processors and Operating Systems on PCs

Until recently the mainstream desktop PC environment has been a Pentium-compatible 32-bit processor running Microsoft Windows. 32-bit Windows supports up to 4GB RAM, although each running program may use at most 2GB (in practice 1.5GB) for data.

Most new desktop PCs contain a 64-bit[10] CPU with 2 or more processing cores, and can accommodate up to 8GB of RAM. However the usual operating system is still 32-bit Windows—causing the CPU to fall back into a 32-bit compatibility mode, and imposing the memory limitations just described.

These limitations may be surmounted by 64-bit versions (**Win64**) of Windows XP or Vista. Programs compiled especially for 64-bit Windows can access all available RAM (the 2 GB limit is broken). But even older, 32-bit, programs run as fast under Win64 as they did in 32-bit Windows. Indeed Win64 is the best way to simultaneously run several 32-bit programs which each use up to 2GB of RAM. In total, all the RAM on the PC may be used..

The main benefit of 64-bit programs is the greater RAM access – but they offer other potential advantages. For example, the speed penalty for using double-precision is reduced (but not removed). There are more high-speed registers, which a very few especially-tuned programs may use. But in general we would not expect that a program originally compiled for a 32-bit CPU would run faster when re-compiled for Win64. Indeed, the 64-bit program will require a little more RAM than its 32-bit counterpart.

Possible compiler/Windows combinations for use with GEMPACK are:

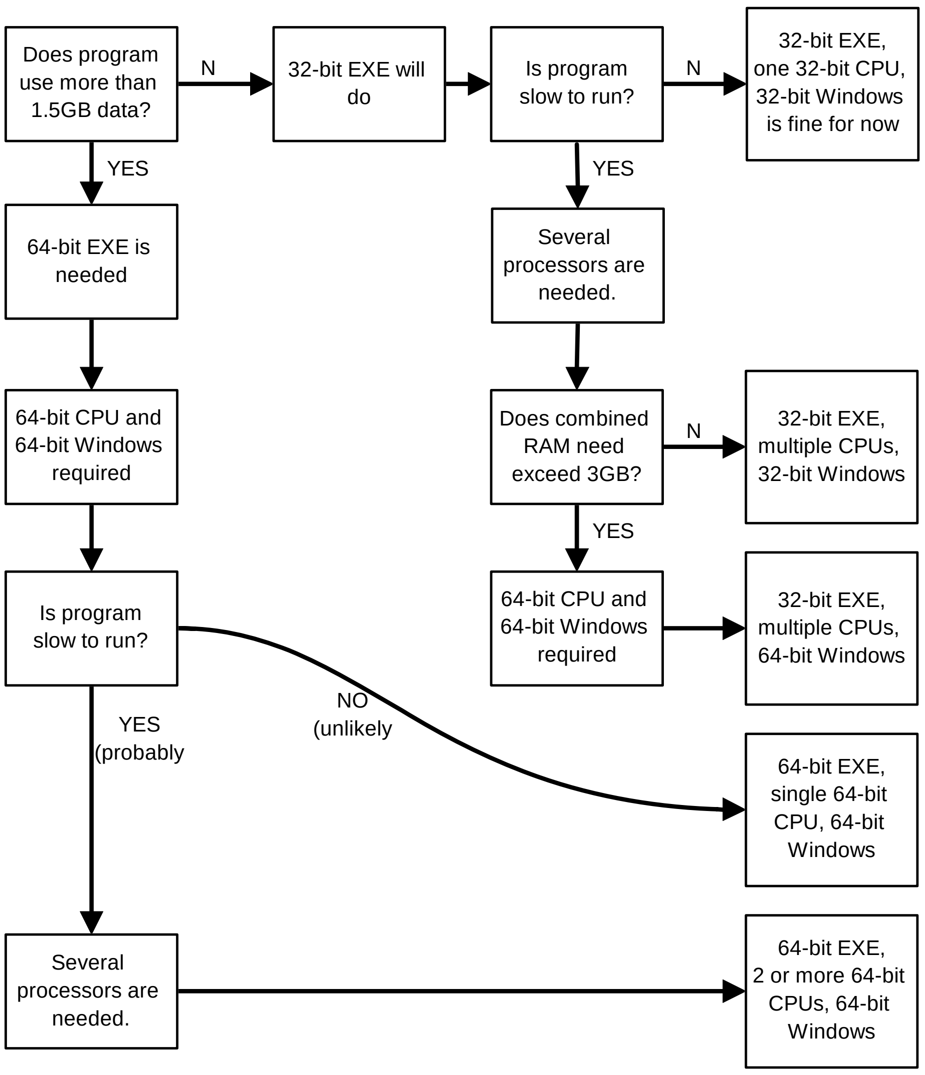
	32-bit Windows (Win32)	64-bit Windows (Win64)
32-bit Intel or Lahey compiler	standard configuration	works well with multi-core CPU, but each program can access only 2GB memory
64-bit Intel compiler	**not allowed**: 64-bit programs will not run on Win32	required if model needs more than 2GB

The flow chart below will help you decide if you should consider moving to the 64-bit Intel compiler now. If 32-bit is still ok for you, you can continue to use the Lahey LF95 compiler or switch to the Intel 32-bit compiler.

Some more advice relating to the above matters may be found at:

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

[10] We refer here to the x86-64 or AMD64 instruction set architecture designed by AMD and copied by Intel (who have named it variously Intel64, Yamhill, Clackamas Technology, CT, IA-32e, or EM64T). We are **not** talking here about Intel's Itanium 64-bit (IA64) processor, or the special version of Windows (Server 2003 Itanium) designed for it. The Itanium is internally quite different from the Pentium and so existing 32-bit programs (compiled for 32-bit Windows) can run only relatively slowly, using an emulation layer. The difficulty of using older programs has restricted the Itanium to a niche market.


Does program use more than 1.5GB data?
N
32-bit EXE will do
Is program slow to run?
N
32-bit EXE, one 32-bit CPU, 32-bit Windows is fine for now
YES
YES
64-bit EXE is needed
Several processors are needed.
64-bit CPU and 64-bit Windows required
Does combined RAM need exceed 3GB?
N
32-bit EXE, multiple CPUs, 32-bit Windows
YES
Is program slow to run?
64-bit CPU and 64-bit Windows required
32-bit EXE, multiple CPUs, 64-bit Windows
NO (unlikely
YES (probably
64-bit EXE, single 64-bit CPU, 64-bit Windows
Several processors are needed.
64-bit EXE, 2 or more 64-bit CPUs, 64-bit Windows

4.1 Memory Limits with Release 10 of GEMPACK

Even with the 64-bit Intel compiler, there are limits to the amount of memory that programs from Release 10 of GEMPACK can access.

In Release 10, dimensions of arrays are 4-byte integers. That means that any array dimension is limited to size 2,147,483,647 (that is, 2^31-1) since this is the largest 4-byte integer – see section 7.11.3.

Large models use large amounts of memory mainly because they need to increase the parameters MMNZ, MMNZ1 and MMNZ2 which determine the amount of memory used during LU decomposition (see sections 12.1 and 13.3 of GPD-3). Each of these parameters determines the dimension of a vector array. Each such array has size (in bytes) equal to 4 times the size of the relevant parameter. If each of MMNZ, MMNZ1 and MMNZ2 is equal to the largest dimension 2,147,483,647, each array will occupy 8GB (8 gigabytes) of memory. Hence the total amount of LU decomposition memory which can be allocated by a Release 10 GEMPACK simulation program is about 24GB (3 arrays each limited to 8GB). Since the rest of the memory needed, even for a very large model, is likely to be at most 4-8GB, that means the largest amount of memory a Release 10 simulation program can realistically need is about 32GB. Of course, if you want to run with 2 servants, the master and each servant may need this amount of memory. And if you want to run several simulations at the same time (perhaps under RunMONASH) that will use more memory.

Because of the 4-byte integer limit, if you specify an integer larger than 2,147,483,647 in a "start with MMNZ =" statement (see section 13.3 of GPD-3) in a Command file, that will produce a fatal error. For example

`start with MMNZ = 3000000000 ;  !` 3000 million – **not allowed**

For Release 11, we expect to change the dimensions of some large arrays (including the declarations of MMNZ, MMNZ1 and MMNZ2) to be 8-byte integers. Then, with the Intel 64-bit compiler, any single dimension can go up to the massive size of 2^63-1 which is equal to 9,223,372,036,854,775,807. It may take a while for computers to have sufficient memory to realise arrays of such size.

4.1.1 Other Size Limits

Dimensions for all arrays in Release 10 are 4-byte integers. Therefore the following sizes are all limited to a maximum of 2,147,483,647:

- The number of components in a Coefficient or Variable.
- The total number of columns in the Equations Matrix (see section 2.13 of GPD-1) [that is, the total number of components of all variables in the condensed system.]
- The total number of rows in the Equations Matrix [that is, the total number of components of all Equations in the condensed system]
- The total number of components in all condensed plus backsolved variables.
- The total number of entries in the Equations Matrix (see section 2.13.1 of GPD-1), or in the LHS Matrix (see section 2.13.2 of GPD-1) when forming up equations.
- The total number of nonzeros when calculating the LU decomposition (see section 12.1 of GPD-3).
- the value used in a "start with MMNZ" statement (see above).

4.1.2 Limits for GEMSIM

GEMSIM puts the values of all real Coefficients into a single array called RCMEM. Even with the Intel 64-bit compiler, the size of this array is limiter to 2,147,483,647 (that is, 2^31-1). Similarly, GEMSIM puts the values of all Integer Coefficient into a single array called ICMEM which has the same size limit. If you model requires more memory than allowed by these limits, you will see an error message which refers to RCMEM or ICMEM.

This limit only applies to GEMSIM, not to TABLO-generated programs.

CHAPTER 5

5 Re-using Pivots (MA48)

This section only applies when the solve time option **Re-using pivots** (see section 12.3 of GPD-3) has been chosen and when the MA48 routines are being used to solve the model (see section 12.1.1 of GPD-3).

Under Release 9 (and earlier) of GEMPACK, the pivot re-use option was available (and switched on by default); it sometimes reduced solution time considerably. In cases where pivot re-use was unhelpful, it could be switched off by including the following line in your Command file.

`NRP = yes;` ! yes means do not re-use pivots

For Release 10, the pivot re-use strategy has been improved to greatly increase the likelihood that it will indeed speed up solutions. If your CMF files contain lines like the above (deactivating pivot re-use), you may find it worthwhile to comment them out (put a '!' at the line start) to see if the new pivot re-use strategy helps.

Below we describe the Release 10 pivot re-use improvements in detail (perhaps more detail than you will need on a first reading).

5.1 New Pivot Re-use Strategy

The MA48 linear equation solver exploits sparsity in the equation system. Only non-zero coefficients are stored and manipulated. Re-use of pivots assumes that the pattern of sparsity is the same at each step. Therefore pivot re-use becomes difficult if an initially-zero coefficient becomes non-zero and breaks the sparsity pattern during the calculation.

When using the MA48 routines, re-use of pivots at one step is only possible if the entries on the left hand side were all present when pivots were last analysed (which is done in a call to MA48A on a previous step). When re-using pivots works, the elapsed time for the step is usually about 60% less than would have been the case if pivots had not been re-used. [If pivots can be re-used, only routine MA48B is called. If re-using pivots is not possible, MA48A must be called, then MA48B.] [11]

Under Release 9 (and earlier) of GEMPACK, each time MA48A is called, pivot information from it is stored. On subsequent steps on the same multi-step calculation, the left-hand-side entries are compared with those sent to MA48A when it was last called. Re-use of pivots on the current step is only possible if every LHS entry on the current step was present the last time MA48A was called. For each separate multi-step calculation, LHS entries for step 2 are compared with those from step 1 of the first multi-step calculation.

We noticed that, in many cases, re-use of pivots was not happening as often as was desirable. Further investigation showed that, in some cases, LHS entries were flip-flopping in and out of the LHS. That is, for certain positions in the LHS Matrix, there is a nonzero entry in some steps and a zero entry (which was not being stored) in other steps. If there was a zero entry the last time MA48A was called and there is a nonzero entry on a subsequent step, re-use of pivots is not possible.

Unless you have opted for memory sharing (see section 4.2 of GPD-5), we have put into place a new strategy which works as follows. [12]

[11] By default, the routines called MA48AG and MA48BG are used – see section 5.2 of GPD-5.

[12] The new re-use of pivots strategy described here may increase MMNZ etc considerably. That is undesirable when memory sharing is in force since then, presumably, memory (including for nonzeros) is in short supply. That is why we do not use the new strategy for re-use of pivots when memory sharing has been selected. More specifically, the "always increase set" strategy is not used when memory sharing is in force. But, at

1. Whenever an attempt to re-use pivots fails because there is a LHS entry which was not present the last time MA48A was called, we ensure that an entry is always sent in this place (a zero entry if necessary) in every subsequent step whenever it is necessary to call MA48A. We think of this as

   always increase the set of positions of entries on the LHS which are sent to MA48A.

2. For step 2 of the second multi-step calculation, we do not compare the LHS entries with those from step 1 of the first multi-step calculation (as was done in Release 9) but rather compare with those from the last step of the first multi-step calculation. This ensure that the set of positions sent to MA48A always increases through the calculation, even from one multi-step calculation to the next. Similarly for the second step of the third multi-step calculation.[13] [14]

3. The Harwell routines MA48A and MA48B rely on permuting the LHS Matrix to a block upper-triangular form. Suppose that all "new" entries encountered (that is, entries which were not present the last time MA48A was called) would be above or in the diagonal blocks of the permuted LHS. Then the program (GEMSIM or the TABLO-generated program) is able to **modify the pivot information** saved from the previous step to pretend that these "new" entries were present at that step.[15] Then the program is able to re-use pivots by calling MA48B (bypassing the slower call to MA48A which would have been necessary in Release 9). There are two cases.
   (a) If all "new" entries are above the diagonal blocks, the normal fast call to MA48B is made. That should proceed just as quickly as if these "new" entries were not present.
   (b) If one or more "new" entries are in the diagonal blocks, a slightly slower call to MA48B is made since the pivot order may need to be changed for numerical reasons.[16]
   Below we refer to this as the

   modify pivots and re-use if all new are above or in diagonal blocks strategy.

There are Command file statements which allow you to turn these strategies on and off (see section 5.2).[17]

---

present, we do allow MA48 to modify and re-use pivots if all new entries are above (not on) the diagonal since that usually only increases MMNZ by a very small number.

[13] This makes arithmetic on the second and third multi-step calculations potentially dependent on the first multi-step calculation. So, if using one or more servants (see chapter 2), the exact results may be slightly different from when the simulation is run normally (without servants). For example, in the GTMSEUFG.CMF simulation described in one of the Examples in this section, re-use of pivots may fail on the second pass of any multi-step calculations carried out by a servant
If debug option 160 is set, for step 2 of all multi-step calculations, the LHS is compared with that from the first step of the first multi-step calculation (as was done in Release 9 and earlier).

[14] The LU decomposition is always recalculated from scratch at the start of each subinterval (see section 5.5). In keeping with the "always increase" idea, the set of entries sent to MA48A at the start of any subinterval includes (as zero entries if necessary) any entries encountered on any step of the previous subinterval (though this did not happen before Release 10).

[15] In fact the software pretends that there were entries equal to zero in the relevant places on the previous step (when the pivot information obtained by the Harwell MA48 routines was last stored). This is a non-trivial enhancement of the capability of the MA48 routines.

[16] This slower call to MA48B is made with JOB=1 (whereas the faster call is made with JOB=2). We have noticed that these slower calls may require a significant increase to MMNZ values (hence increasing memory requirements) in some cases. For that reason, if you are using memory sharing (see section 4.2 of GPD-5), re-use of pivots is only implemented when all new entries are above the diagonal in which case the faster JOB=2 call is made to MA48B – this call does not require an increase in MMNZ.

[17] See section 5.2 below if you want to revert to the Release 9 re-use of pivots strategy.

Our testing indicates that the changes above from the Release 9 re-use of pivots strategy will

- result in faster solving in many cases,
- will rarely slow things down, and
- will not require too much extra memory.[18]

5.1.1 Examples

Example 1

Suppose that you are doing an Euler 2,4,6-step calculation. Suppose that the entry in row 10, column 20 of the LHS Matrix is nonzero at step 1 but zero at step 2. And suppose that re-using pivots fails at step 2 because the entry in row 15, column 23 is zero at step 1 but was non-zero at step 2. Then the software must call MA48A at step 2. When the software sends the LHS Matrix to MA48A at step 2, a zero entry is sent in row 10, column 20. Then, if this entry flip flops back to be non-zero at a subsequent step, that will not cause re-using pivots to fail. This is an example of the "always increase set of positions sent to MA48A" strategy described above.

Example 2 – GTMSEUFG.CMF Simulation

Consider the GTMSEUFG.CMF simulation with the GTAP61 model (supplied with the GEMPACK examples). This is a Gragg 2,4,6-step calculation.

When run with the Release 9 strategy (using a TABLO-generated program and the LF95 compiler with optimisation turned off[19]), we find that re-using pivots succeeds only in step 2 of the third, 7-pass, Gragg calculation. Re-use of pivots fails at all other steps.[20] The details are as follows.

- On pass 2 of the first multi-step calculation, the entries corresponding to equation TINCRATIO("EU"), variable pm("UnskLab","EU") and to equation VGDP_r("ROW"), variable pp("Mnfcs","ROW") are nonzero whereas they were zero on the first pass.
- On pass 3 of the first multi-step calculation, the entries corresponding to equation VGDP_r("SSA"), variable pp("Svces","SSA") and to equation VGDP_r("EU"), variable pp("Svces","EU") are nonzero whereas they were zero on the second pass.

[18] We have found a small number of simulations in which modifying and re-using pivots when some new entries are in the diagonal blocks of the permuted matrix requires an unacceptably large increase in MMNZ. If that happens, you can use the Command file statement

```
pivots modify MA48 = above ;
```

(see section 5.2 below) to restrict re-use to the case where all new entries are above the diagonal of the permuted matrix. [That normally requires only very modest increases in MMNZ.]

[19] Changing the compiler and/or optimisation settings can change what happens when we attempt to re-use pivots, since many of the nonzero entries which potentially prevent the re-use of pivots are very small numbers produced by subtracting two nearly equal numbers. Compiler settings can affect whether such differences are tiny or exactly zero. For a concrete example, see the footnote which discusses the pm terms in equation TINCRATIO later in this Example.

[20] However, using GEMSIM instead of a TABLO-generated program (and still LF95 with optimisation turned off), re-use of pivots succeeds in all but steps 4 and 7 of the third multi-step calculation. [These results obtained using Release 9.0-003 of GEMPACK with Command file GTMSEUFG-LIKEREL9.CMF which writes to the terminal at each step the values of Coefficients PTAX and C00336KP which give the entries in the VGDP_r,pp and TINCRATIO,pm submatrices respectively. The Release 10 results were obtained using Command file GTMSEUFG-REL10.CMF]

- On pass 2 of the second multi-step calculation, the entry corresponding to equation VGDP_r("ROW"), variable pp("Mnfcs","ROW") is nonzero whereas it was zero on the first pass of the first multi-step calculation.
- On pass 3 of the second multi-step calculation, the entries corresponding to equation TINCRATIO("SSA"), variable pm("SkLab","SSA") and to equation VGDP_r("SSA"), variables pp("Svces","SSA") and pp("Mnfcs","SSA") plus 3 other entries in the VGDP_r,pp submatrix are nonzero whereas they were zero on the second pass.
- On pass 4 of the second multi-step calculation, the entries corresponding to equation TINCRATIO("EU"), variable pm("UnskLab","EU") plus two entries in the VGDP_r,pp submatrix are nonzero whereas they were zero on the third pass.
- On pass 5 of the second multi-step calculation, the entries corresponding to equation TINCRATIO("ROW"), variable pm("Capital","ROW") plus 3 entries in the VGDP_r,pp submatrix are nonzero whereas they were zero on the fourth pass.
- On pass 3 of the third multi-step calculation, the entries corresponding to equation TINCRATIO("EU"), variable pm("UnskLab","EU") plus 3 entries in the VDGP_r,pp submatrix are nonzero whereas they were zero on the first pass of the first multi-step calculation (which is where the relevant pivots were calculated since re-use of pivots succeeded on step 2 of this third multi-step calculation).
- On pass 4 of the third multi-step calculation, the entries corresponding to equation TINCRATIO("SSA"), variable pm("SkLab","SSA") and to equation VGDP_r("SSA"), variable pp("Food","SSA") are nonzero whereas they were zero on the third pass.
- On pass 5 of the third multi-step calculation, 4 entries in the VGDP_r,pp submatrix are nonzero whereas they were zero on the fourth pass.
- On pass 6 of the third multi-step calculation, the entries corresponding to equation TINCRATIO("SSA"), variable pm("SkLab","SSA") plus 3 entries in the VDGP_r,pp submatrix are nonzero whereas they were zero on the fifth pass.
- On pass 7 of the third multi-step calculation, the entries corresponding to equation TINCRATIO("EU"), variable pm("UnskLab","EU") and to equation VGDP_r("EU"), variable pp("Food","EU") are nonzero whereas they were zero on the sixth pass.

You can see the flip-flopping that is happening. [For example, TINCRATIO("EU"), pm("UnskLab","EU") reappears several times.] You can also see that the two submatrices TINCRATIO,pm and VGDP_r,pp keep occurring – these are the only submatrices in which "new" entries appear.[21]

With the Release 10 strategy, new entries in the two submatrices TINCRATIO,pm and VGDP_r,pp appear (as before). It turns out that each new one which appears is above the

[21] The `pm` term in the `TINCRATIO` equation is

```
    (all,r,REG)   SUM[i,ENDW_COMM, PTAX(i,r)*pm(i,r)]
```

Here PTAX(i,r) is calculated as VOM(i,r) – VOA(i,r). The VOM and VOA values are calculated by adding several values together.

In the simulation in question, the values of PTAX(i,r) should be exactly zero for i in ENDW_COMM since there are no taxes on factors in the base data or in this simulation. However, in practice, sometimes the PTAX values are calculated to be exactly zero and sometimes to be small nonzero values. This is typical of the sorts of rounding that occur when several arithmetic operations are carried out on a computer.

Similarly, the pp terms in equation VGDP_r [these terms enter during condensation – the coefficient of pp(i,r) in Equation VGDP_r(r) is what TABLO calls C00336(i,r)] should be exactly zero – but again in practice the rounding that occurs means that they are sometimes zero and sometimes small nonzeros.

diagonal in the permuted LHS matrix [case 3(a) above].[22] So MA48A never needs to be called and the program is able to re-use pivots at each step.

This example shows the advantages of the Release 10 strategy, notably

- the "always increase the set" idea and
- the "modify and re-use if all new are above or in diagonal" enhancement.

Example 3 – USAGE-ITC

This relates to a 6-step Euler simulation with the large USAGE-ITC model of the USA, using a TABLO-generated program made with the Intel 32-bit compiler using O2[23] optimisation.[24]

We found that pivots were not re-used at all with the Release 9 strategy.

However all new entries at steps 2-6 would be above or in the diagonal blocks of the permuted matrix. Thus, with the Release 10 modify and re-use strategy, pivots are re-used in all of steps 2-6, which reduces the elapsed time from about 56 minutes to about 29 minutes.

Indeed, at step 2 there are new entries in and above the diagonal (several thousand above and about 5 in). So a slower call (JOB=1) is made to MA48B at step 2. Unfortunately that call increases MMNZ by over 15 percent (from about 42 million to about 48 million).

At steps 3-6, all new entries are above the diagonal blocks.

Using the alternative strategy of only modifying and re-using pivots in such a case (that is, calling MA48A to recalculate pivots if there are any new entries in the diagonal blocks) we found that re-using pivots was still possible in all of steps 3-6. The elapsed time using this strategy is still about 31 minutes (still acceptable). In this case, MMNZ had to be increased to about 46 million in step 2 (less than the 48 million needed above) and then does not need to be increased further in the remaining steps.

Similar issues and times result when using the Intel 64-bit compiler.

However interestingly and perplexingly, using the LF95 compiler and the Release 9 strategy for re-use of pivots, pivots are re-used in all of steps 3-6. That shows that whether or not pivots are re-used can change dramatically depending on the compiler. It can also change if

[22] On pass 2 of the first multi-step calculation, two new entries corresponding to equation TINCRATIO("EU"), variable pm("UnskLab","EU") and to equation VGDP_r("ROW"), variable pp("Mnfcs","ROW") appear.
On pass 3 of the first multi-step calculation, one new entry corresponding to equation VGDP_r("SSA"), variable pp("Svces","SSA") occurs (and it is above the diagonal of the permuted LHS). There is also a nonzero entry corresponding to equation VGDP_r("EU"), variable pp("Svces","EU"). Although the corresponding entry was zero in pass 2 it was nonzero on pass 1 so the "always increase the set" strategy means that it is not considered a new entry on pass 3.
After that, it turns out that new entries are encountered only on
pass 3 of second multi-step calculation. [New entry corresponds to equation VGDP_r("SSA") and variable pp("Mnfcs","SSA").]
pass 5 of second multi-step calculation. [New entry corresponds to equation VGDP_r("SSA") and variable pp("Food","SSA").]
pass 4 of third multi-step calculation. [New entry corresponds to equation TINCRATIO("SSA") and variable pm("Land","SSA").]
Each is added to the LHS and to the set of pivots. Since each one happens to be above the diagonal in the permuted LHS, re-use of pivots occurs at each step.

[23] O1 optimisation is now the default with GEMPACK for the Intel compiler – see "compiler options" in chapter 7 of GPD-6.

[24] This is from MONUSA.TAB dated 13 March 2006 and Ken calls the Command file BI32FSB-1992.CMF. Specifically BI32FSB-1992-REL10.CMF for the Release 10 results, BI32FSB-1992-REL10-ABOVE.CMF for the Release 10 results when re-use of pivots is only done if all new entries are above the diagonal and BI32FSB-1992-LIKEREL9.CMF for seeing the location of the entries which prevent re-use of pivots with Release 9.

different optimisation settings are made with the same compiler. All these things affect whether some value which should theoretically be exactly zero ends up in practice being exactly zero or not.

5.1.2 Another Possible Strategy

We also experimented with the following strategy.

Whenever a "new" entry is encountered (that is, an entry which was not present the last time MA48A was called), we add not just the entry itself. Rather we send all relevant entries in the relevant submatrix (this is the submatrix determined by all equations in the relevant equation block and all endogenous components of the variable) to the set of entries sent to MA48A.[25] Again we send zero entries if necessary to follow the "always increase" idea described above.[26] We call this strategy

add whole submatrix.

This strategy works well with moderately sized models.[27]

But with large models (such as the USAGE-ITC model – see example 3 above), the submatrices in question can be very large which would require the addition of tens or hundreds of thousands of new entries to the set of entries sent to MA48A. Such large increases are not practical since they can easily cause the program to exceed the memory limits. So we have abandoned this "add whole submatrix" strategy.

5.2 New Command File Statements for Pivot Re-use

We have added new Command file statements which allow you to turn on or off some of the re-use of pivots strategies described in section 5.1.

The first one of these relates to the "always increase" strategy described in point 1 in section 5.1.

```
pivots keep adding MA48 = yes|no ;
```

- Default is "yes" unless memory sharing is in force when default is "no".

The next one relates to the "modify pivots" strategy described in point 3 in section 5.1.

```
pivots modify MA48 = no|above|yes ;
```

- "no" means do not modify pivots (even if all "new" entries are in or above the diagonal of the permuted LHS Matrix).
- "above" means only modify pivots if all "new" entries are above (not in) the diagonal of the permuted LHS Matrix.
- "yes" means modify pivots if all "new" entries are in or above the diagonal of the permuted LHS Matrix.
- Default is "yes" unless memory sharing is in force (see section 4.2 of GPD-5) or you are using the original MA48 routines (see section 5.2 of GPD-5), when default is "above".

[25] Sometimes we try all positions in the submatrix. Other times the software is able to identify a subset of the submatrix in which any entries must lie. In the latter case, the software just adds positions from this subset (since to add all would be wasteful of memory). This is what we mean by "relevant entries" in the sentence in the main text to which this footnote is attached.

[26] Adding the whole submatrix is like having "iz1 = no ;" (see section 12.5 of GPD-3) for just that submatrix.

[27] With the GTMSEUFG.CMF example described earlier, the "add whole submatrix" allowed re-use of pivots for steps after step 2 of the first multi-step calculation since all entries of the two offending submatrices were added during step 2 of that calculation.

We have found a small number of simulations in which modifying and re-using pivots when some new entries are in the diagonal blocks of the permuted matrix requires an unacceptably large increase in MMNZ (see section 5.3). If that happens, you can use the Command file statement

```
pivots modify MA48 = above ;
```

to restrict modifying pivots to the case where all new entries are above the diagonal of the permuted matrix. [That normally requires no increase in MMNZ, or at the worst only a very modest increases in MMNZ.]

If you put

```
pivots modify MA48 = above ;
```

in your Command file and a "new" entry is in one of the diagonal blocks, you will see the message

```
 [Not re-using pivots since are "new" entries in
  the diagonal blocks - see section 5.2 of GPD-9.]
```

Note that the two strategies "always increase set" and "modify and re-use" in points 1 and 3 of section 5.1 are independent of each other. It is perfectly acceptable to decide to use one without using the other (though both are usually used by default).

To revert to the Release 9 re-use of pivots strategy, include the statements

```
pivots keep adding MA48 = no ;
pivots modify MA48 = no ;
```

in your Command file.

5.3 Modifying Pivots When New Entries are In the Diagonal Blocks

As indicated in point 3 in section 5.1 above, if some new entries are in the diagonal of the permuted matrix (but none below the diagonal), modifying and re-using pivots is achieved via a slower call to MA48B with JOB=1. You will see

```
  [Calling MA48B with JOB=1 - partial re-use of pivots]
```

in the LOG file when this happens.

We have noticed that these JOB=1 calls may require a significant increase to MMNZ values (hence increasing memory requirements) in a few cases.[28] And, in those cases (and perhaps others), there may not be any CPU saving from re-using pivots.

When doing one of these JOB=1 re-use of pivots calls, if routine MA48BG asks to reallocate the MMNZ arrays because the current MMNZ is not large enough, we have decided it is best to abandon the attempt to re-use pivots. Instead the software recalculates the LU decomposition from scratch. If that happens you will see a message something like the following:

```
    Since JOB=1, will redo LU from scratch rather than increasing MMNZ.
```

Later we may try to be more sophisticated and abandon the re-use attempt in this JOB=1 case if we detect that

- CPU time has already exceeded that required for doing LU from scratch on previous steps, or

[28] One example is GTEMSIM1.CMF in our TESTGP suite GTEM1-GPT.ZIP. When this is used with the Intel 32-bit compiler and /O2 optmisation for MA48 and /O1 for the TABLO-generated program, re-use with JOB=1 in step 5 of the 5-pass Euler computation results in a massive increase in MMNZ. The increase is so massive that we are not able to solve it within the 2GB allowed under 32-bit Windows. With the strategy described in the text, as soon as MA48BG detects the first request to increase MMNZ, the software bales out of the re-use attempt and does the LU decomposition from scratch.

- MMNZ has already been increased by more than say 10% above the value required when doing LU from scratch on previous steps.

If you wish to experiment, you can turn off the JOB=1 re-use by including the statement

```
pivots modify MA48 = above ;
```

(see section 5.2 above) in your Command file.

5.4 Never Modify Pivots if a Triangular Block Would Become Non-Triangular

We have found a small number of examples in which there is a new entry below the diagonal in a diagonal block which was (upper) triangular. An upper triangular block has a particularly simple LU decomposition – just invert all the entries on the diagonal.

But when an entry goes below the diagonal, the LU decomposition of the resulting block may be very slow because it then requires lots of memory.

We have found a few examples in which a "new" entry below the diagonal in a formerly upper triangular diagonal block requires a huge increase in MMNZ if we attempt to modify and re-use pivots with JOB=1 (see section 5.3 above). Because some of the increases in MMNZ are massive[29], and because the JOB=1 re-use of pivots may be a little problematical, the software never attempts to modify and re-use pivots in such a case. Instead pivots are recalculated from scratch.

If the above occurs, you will see the message

```
 [Not re-using pivots since otherwise triangular diagonal
   block number xxx would be no longer,
    triangular - see section 5.4 of GPD-9.]
```

5.5 Start of Second and Subsequent Subintervals

If you are using several subintervals, the software always calculates the LU decomposition from scratch during the first step of each subinterval. The LU decomposition is redone from scratch each subinterval since numerical accuracy might slip if re-use of pivots continued throughout a long and numerically demanding simulation (which may be the case since you are using more than one subinterval).

In the first step of the second and subsequent subintervals, the software does however attempt to continue the "always increase the set of entries on the LHS which are sent to MA48A" idea explained above. This is done by comparing the set of entries on the LHS with those in the set last sent to MA48A. If necessary, zero entries are added to the set sent to MA48A.[30]

[29] One example is in the 1A5R.CMF simulation with model G5BTRQ – see section 16.8.4.3 of GPD-3. With the Intel 32-bit compiler and certain optimisation settings, MMNZ about 50,000 was ok on step 1 but, when a new entry made an upper triangular block non-triangular, suddenly MMNZ had to be increased to over 4 million in order to re-use pivots.
In a second example with the large USAGE model of the USA, MMNZ of about 17 million had to be increased to about 32 million. We are grateful to Ashley Winston who reported this problem – this report made us re-think the general strategy to avoid attempting to re-use pivots when a new entry makes an upper triangular block become non-triangular.

[30] Of course, this comparison of entries does take some CPU time. We expect that the subsequent saving by re-using pivots will usually save CPU time overall.

CHAPTER 6

6 Enhancement – Sparse Header Arrays

Header Arrays containing real numbers (types RE and RL) can now be written in sparse form[31]. This is normally done only if at least 60% of the values are zero. When arrays are very sparse (that is, contain a high percentage of zero values), this can reduce the size of the Header Array file considerably. Arrays of integers and character strings are never written in sparse form, nor are 2-dimensional arrays of type 2R. [See section 3.1.1 of GPD-4 for more about array type.]

For example, the main data file for the USAGE model of USA is reduced from about 55 Mb to about 6 Mb by writing the relevant arrays in sparse form.

When a program writes a Header Array in sparse form, it just writes the nonzero values in the array and the corresponding positions (integers) to the file.

For example, consider a 4x3 array in which the only nonzero entries are the (1,2) entry equal to 6.102 and the (4,3) entry equal to 7.366. Then this array would now normally be written in sparse form. On the file would go

- the array containing the two nonzero values [6.102, 7.366], and
- the array indicating the positions of these values in the whole array [5, 12]. [The positions are calculated by varying the first index fastest, and so on, as for component number as explained in section 5.3 of GPD-3.]

If you use recent GEMPACK Fortran or Windows programs to write sparse header arrays, you must also use appropriately recent versions of all relevant programs which can read such files. For example, the Release 9.0 (April 2005) versions of SEEHAR, CMPHAR, GEMSIM, ViewHAR, ViewSOL and AnalyseGE cannot read Header Array files which contain sparse headers. Accordingly you must make sure that you are using recent versions of these programs which can read sparse headers.

For Release 10.0, the programs will only write sparse headers if you specifically request them to (see section 6.1 below). Then the program decides whether to write each header in sparse form. The programs will write arrays of types RE and RL in sparse form if at least 60% of the values in the array are zero.

The default is that the programs do not write sparse arrays.

For future GEMPACK releases, we will probably change the default so that then the programs will write sparse headers whenever the array is sparse, unless you specifically request them **not** to.

6.1 Writing Sparse Headers

If you are preparing a Header Array file to send to a friend or colleague who does not have the recent GEMPACK software which can read sparse header arrays, you will want to make sure that you do not write any arrays in sparse form.

6.1.1 GEMPACK Fortran Programs

Versions of the GEMPACK programs beginning with Release 9.0-002 (November 2005) can all read header arrays written in sparse form. [But the Release 9.0 (April 2005) versions of these programs cannot read such header arrays.]

[31] This section documents changes which were made for Release 9.0-002 (November 2005) of GEMPACK.

For any of the GEMPACK Fortran programs (for example, GEMSIM, TABLO-generated program, SLTOHT, MKHAR, RWHAR), you can ensure that they do write header arrays in sparse form by selecting program option at the first option screen (the screen where you can select such options as LOG, STI, SIF. See section 5.3 of GPD-1.)

WHS Write Headers in Sparse format

when you run the program.

WHS is a new basic program option which is available in all the Fortran programs (Release 9.0-002 or later). It does not show on the options screen when the program runs, but you can always select it. By default, option

–WHS Do not Write Headers in Sparse format

applies. You need to enter **WHS** to turn that off.

In Command files for GEMSIM or TABLO-generated programs (but not for SAGEM), you can include either of the statements

```
WHS = yes|NO ;               ! NO is the default for the present.
```

"WHS = no ;" ensures that the program will not write any header arrays in sparse form.

"WHS = yes ;" asks the program to write header arrays in sparse form when the array is sparse.

6.1.2 Windows Programs

The relevant programs are ViewHAR, ViewSOL and AnalyseGE. RunMONASH etc may also be affected since it sometimes reads Header Array files directly.

Versions of these programs supplied with Release 9.0-002 (November 2005) or later can all read header arrays written in sparse form. [But the Release 9.0 (April 2005) versions of these programs cannot read such header arrays.]

If you want to ensure that these programs do not write header arrays in sparse form, you need to go to the ViewHAR options menu via ***File..Options*** . Then make sure that the option

Use sparse disk storage

is NOT checked. This option in ViewHAR controls the writing of header arrays for the other relevant programs (such as AnalyseGE). ***File..Options..Help*** leads to a link with more information about this.

6.2 Download Programs Which can Read Sparse Headers

Versions of the GEMPACK Windows and Fortran programs produced before about October 2005 cannot read sparse headers on Header Array files. A typical error message might include the phrase:

un-recognized storage format 'SPSE'

If you produce Header Array files which contain sparse headers, or if you obtain such files from others, you will need to download from the GEMPACK website recent versions of the relevant programs (as supplied with Release 9.0-002 or later of GEMPACK) which can read such headers. This includes the Windows programs ViewHAR, ViewSOL, AnalyseGE and RunGEM and includes the Fortran programs SLTOHT, MKHAR, RWHAR, MODHAR and CMPHAR.

6.2.1 If You Use RunGTAP or RunMONASH etc

If you use RunGTAP or RunMONASH etc, you will need to copy versions of the relevant Fortran programs as indicated in sections 12.1 and 12.2 of GPD-5. You will also need to ensure that your RunGTAP or RunMONASH etc has access to recent versions of ViewHAR, ViewSOL etc.

CHAPTER 7

7 Minor Changes/Enhancements

7.1 LF90 Deprecated

The Lahey LF90 Fortran compiler is still supported under GEMPACK Release 10. However, LF90 will not be supported by Release 11. LF90 has the following drawbacks:

- no longer fully supported by Lahey. [This means that several minor problems we have found with LF90 will not be fixed by Lahey. Instead we have to work around them. Also it is possible that LF90 will cease working in a new version of Windows.]
- some problems with long path-names, and folder names containing spaces or Asian characters.
- uses an older Header Array file format (see chapter 15 of GPD-1). Although newer GEMPACK programs can read this format, there is a performance penalty.

Accordingly, we recommend that

LF90 users upgrade as soon as possible to the Intel Fortran compiler (see chapter 3).

The Intel compiler runs on both 32-bit and 64-bit machines. The latter (which allow access to more than 2GB of memory – see chapters 3 and 4) are becoming very common. LF95 has no upgrade path to 64-bit Windows.

7.2 Programs Report Elapsed Time

All programs now report the total elapsed time for the run.

Previously, you could get a record of CPU time by adding the line "CPU=yes;" to your CMF file. That option may still be useful, since "elapsed" and "CPU" measures of time are different:

- "elapsed time" should give the same time as a stopwatch
- "CPU time" is a measure of the time the CPU has worked on your job.

If your GEMPACK program was the only running process, the two times should be the same. However, a typical PC runs many background programs, which all use CPU cycles. So if your program took 10 minutes elasped time, the CPU time might be only 8 minutes.

The precision and consistency of either measurement may vary between compilers and operating systems.

7.3 MMNZ and LU Decomposition via MA48AG

As indicated in section 5.2.1 of GPD-5, compressions (that is, garbage collection) are not done by default. We have found that, for some models, this leads to unnecessarily large values of MMNZ. In some cases, the required values exceed the amount of memory available. Accordingly, if compressions are not turned on via the Command file, we have decided that TABLO-generated programs or GEMSIM will turn on compressions in certain circumstances, as described below[32].

- When the programs increase MMNZ, they usually try to increase more than the minimum (usually about 10% more – the exact values depends on whether or not you have a statement of the form "MA48 increase_MMNZ = " in your Command file – see section 5.2.2 of GPD-5). Sometimes the program is not able to increase MMNZ by the desired amount. If that happens several times, the program will turn on compressions within MA48AG.
- If the program *does* successfully increase MMNZ a number of times, the program will turn on compressions within MA48AG rather than increasing MMNZ.

In such cases, you will see a message of the form

```
 %%INFORMATION. Looks like cannot increase MMNZ etc much more. So,
       instead of increasing MMNZ etc,
    are turning on compression within MA48AG and
       continuing with the same MMNZ etc values.
```

This is the message for the first case above. In the second case, the "Looks like…much more" is replaced by "Have increased MMNZ etc a few times before".

In each case, compressions are only turned on when the routine MA48AG is the routine asking for the increase.

Once compressions are turned on, it is as if the statement

```
MA48 compress = yes ;
```

had been in the Command file.

[32] This section documents changes which were made for Release 9.0-002 (November 2005) of GEMPACK.

7.4 Reporting Arithmetic Errors

This relates to the reporting of arithmetic errors which occur while doing a Formula, Submatrix, Backsolve or Update. GEMSIM and TABLO-generated programs report all instances of the following types of errors.

1. division by zero when this is not allowed.
2. using invalid powers (for example, a negative value raised to a non-integer power).
3. evaluating functions at invalid values (for example, the LOG of a negative number).
4. arithmetic overflow (either real or integer overflow). [Catching integer overflow is new for Release 10 – see section 7.11.][33]

In Release 9, especially if the error occurred during step 2 or later of a multi-step calculation, you had to rerun the simulation

- with "eaa = yes ;" (see section 15.3 of GPD-3) in order to find out exactly which statement the error occurred in,
- and with "dws = yes;" (see section 15.3 of GPD-3) plus appropriate write/xwrite statements in order to find out the values of the relevant Coefficients and/or Variables in the offending statement.

In Release 10, rerunning with "eaa = yes;" and "dws = yes;" is unnecessary since GEMSIM and TABLO-generated programs now write a detailed **arithmetic error report** in the Log file from the run.[34] The arithmetic error report contains

- the type of error.
- the type of statement (Formula, Update etc) and the line number in the TAB file on which the statement which produced the error can be found.
- the name of an automatically generated HAR file from which the values of the Coefficients and Variables in the expression containing the error may be read. The values shown in that HAR file are those for the Coefficients/Variables involved in the relevant step of the multi-step calculation. [Variables can only occur if the error occurs in a Backsolve, Update or (see chapter 2 of GPD-5) in a postsim Formula.[35]]

For the first 3 types of errors listed above (but not type 4– arithmetic overflow), the arithmetic error report also contains

- the values of the active indices (if any) for the first occurrence of the error.
- the full expression being evaluated when the error occurred.
- the values directly involved when the error first occurred (for example, the value of the numerator in the case of a division by zero error, or the value of the argument in the case of an invalid function argument).

[33] Overflow can happen when you multiply 2 large numbers and get a result which is too large to be stored. GEMPACK programs use single-precision reals (4 bytes). The largest positive real number that can be stored is about 3.4*10^38 – see section 7.11.3. As an example, the product (10.0^20)*(10.0^21) will lead to real overflow.

[34] To obtain a LOG file, run the simulation with "log file = yes ;" (see section 2.6 of GPD-3) in your Command file.

[35] When calculating the entries of a Submatrix, the Variable involved has no values – rather its indices indicate what column in the Equations or LHS matrix the entries of the Submatrix go in – see section 2.13 of GPD-1.

- the expression(s) directly involved in the error (for example, the expressions for the numerator and the denominator in the case of a division by zero error). [However, these expressions are not shown when the error occurs doing a submatrix because, at present, we are unable to accurately report these expressions in that case. See Example 3 below.]

Of course it is possible for several errors of the type above to occur in a statement. If so, the report only gives details (values of indices) for the first case. [For example, if a formula ranges over set COM and erorrs occur doing the 4th and 8th elements of COM, details will only be given for the 4th one since that is done first.]

We give several examples below.

If the arithmetic error report indicates that the error occurred
while completing update of Coefficient "<name>"
please also see section 7.4.1 below.

Fixing an arithmetic error problem may involve changing the formula or equation in question to allow for zeros in the data, or involve adding appropriate ZERODIVIDE statements (see section 3.11 of GPD-2) to the TABLO Input file, or it may involve changing the base data and/or closure.

Example 1 – Divison by zero in a Formula

You can engineer a simple division by zero error by modifying the Stylized Johansen model (see GPD-1 section 3.3.3) by inserting the line "FORMULA PC("s2") = 0 ;" immediately after line 116 so that lines 116 and 117 now read

```
FORMULA (all,i,SECT) PC(i) = 1.0 ;
FORMULA PC("s2") = 0 ;
```

The FORMULA & EQUATION beginning on line 131, repeated below, contains a division by PC(i) which will be division by zero when i="s2":

```
FORMULA & EQUATION Comin
   # Intermediate input of commodity i to industry j #
(all,i,SECT)(all,j,SECT) XC(i,j) = DVCOMIN(i,j) / PC(i) ;
```

Running the model with the command file **sjlb.cmf** results in an error. The Log file **sjlb.log** from the run contains the arithmetic error report shown overleaf.

The report tells you the relevant line number in the TAB file, the arithmetic error type (division of nonzero by zero), the value of the numerator, the values of the indices involved, the numerator and denominator expressions (since the operation is division) and the complete expression. The automatically generated HAR file **sjlb-assert-arith-fail.har** contains the values of the two Coefficients **DVCOMIN** and **PC** which appear in the expression that produced the error.

```
     Arithmetic error report
     -----------------------

  %% Error doing formula for coefficient "XC",
     at line 131 in the TAB file.

     Division of a nonzero by zero when this is not allowed.
     This occurred first when:

     nonzero numerator=2.0000000

     (There are 2 active indices)
     index "i" from set "SECT", value "s2" (element 2)
     index "j" from set "SECT", value "s1" (element 1)

     Expression for numerator is:

       DVCOMIN(i,j)

     Expression for the denominator is:

       PC(i)

     Expression being evaluated is:

     (ALL,i,SECT) (ALL,j,SECT) XC(i,j) = DVCOMIN(i,j) / PC(i)

 (Written real array, size 2, header '0001'.)
 (Written real array, size 2x2, header '0002'.)
  [Have written Coefficient values to headers "0001" to "0002"
   of file "sjlb-assert-arith-fail.har".]

     This occurred doing formula for coefficient "XC",
     at line 131 in the TAB file.

  [See section 7.4 of GPD-9 for documentation about
     arithmetic error reports.]

     If you have a log file search for "Arithmetic error
       report" to locate the start of the report.

     End of arithmetic error report
     ------------------------------
```

Example 2 – SQRT in a Formula

Consider the Formula

```
Formula (all,c,COM)
  COEF1(c) = SUM(i, IND, C2(c,i) + SQRT[C3(c,i)+C4(c)]) ;
```

Suppose that, on step 3 of a multi-step calculation, the values are such that the expression inside the SQRT is negative (which is not allowed). Then the program will report an error. Amongst other things it will tell you that the expression involved as the function argument is

```
C3(c,i) + C4(c)
```

The values of the indices for which that expression is negative (and the negative value) will be indicated. The values of all RHS Coefficients **C2, C3** and **C4** during step 3 (all of these values, not just those for the indices for which the error occurred) will be shown in the relevant assert-arith-fail HAR file.

If you use **BADSQRT.TAB** and **BADSQRT.CMF** supplied with GEMPACK, you will be able to see the arithmetic error report.

Example 3 – Division by zero in a Submatrix

We now consider a more realistic and slightly more complex example based on an extract from ORANIG03.TAB (ORANIG03.TAB is the 2003 version of the ORANI-G model, see section 1.8 GPD-1). Consider the following modified code extract from ORANIG03.TAB. This TAB file, called **OG03EXT2.TAB**, is in the standard examples supplied with GEMPACK.

```
Coefficient
 (all,i,IND) V1LAB_O(i)   # Total labour bill in industry i #;
Formula
 (all,i,IND) V1LAB_O(i) = sum{o,OCC, V1LAB(i,o)};
Equation
 E_x1lab   # Demand for labour by industry and skill group #
  (all,i,IND)(all,o,OCC)
   x1lab(i,o) = x1lab_o(i) - 0.3*[p1lab(i,o) - p1lab_o(i)];
 E_p1lab_o # Price to each industry of labour composite # (all,i,IND)
    p1lab_o(i) = sum{o,OCC, (V1LAB(i,o)/V1LAB_O(i))*p1lab(i,o)};
Backsolve p1lab_o using E_p1lab_o ;
```

Here OCC has just two elements, "skilled" and "unskilled". The "Dwellings" industry does not use any labour, in particular V1LAB("Dwellings","skilled")=0, V1LAB("Dwellings","unskilled")=0 and so in aggregate V1LAB_O("Dwellings")=0. In fact, our example is artificial since the data (ozdat867.har) supplied with the ORANIG03.TAB example uses a "TINY" number instead of exactly 0 for the value of V1LAB("Dwellings","skilled") and V1LAB("Dwellings","unskilled"). For the purposes of this example we suppose that these data are true zeros. Given this is the case, clearly this will give rise to a division by zero arithmetic error when Equation E_p1lab_o is evalutated. However, the backsolve statement in the extract specifies that p1lab_o will be substituted out using Equation E_p1lab_o. Thus we might expect a division by zero arithmetic error to arise (post substitution) from Equation E_x1lab. This is indeed the case; the resulting arithmetic error report is shown in the log file below.[36]

Notice that the reported active indices are "o", "i", and "XXO1". XX01 is a system generated index variable, automatically generated as a result of substituting the SUM using index "o" from Equation E_p1lab_o into Equation E_x1lab which already involved an occurrence of an "o" index. Moreover, the arithmetic error report tells you that the expression being evaluated (when division by zero occurred) is

```
(ALL,o,OCC) (ALL,i,IND) 0.3 * p1lab(i,o) +
   SUM(XX01,OCC,-0.3 *
V1LAB(i,XX01) / V1LAB_O(i) * p1lab(i,XX01))
```

which clearly does not occur in the original ORANIG03.TAB code extract in Equation E_x1lab. However it can easily be verified that this expression arises as a result of the backsolve. Examination of the associated Information (*.inf) file reveals that a substitution was made in Equation E_x1lab. The lesson here is to take care when reading an arithemtic error report arising from a condensed system.

[36] To see this error report, run TABLO on OG03EXT2.TAB, compile and link if using Source-code GEMPACK, and then carry out the simulation using Command file OG03EXT2.CMF. Note that the backsolve mentioned in the text is included in OG03EXT2.TAB. These files OG03EXT2.TAB and OG03EXT2.CMF are supplied as standard examples with GEMPACK.

```
%% Error doing  submatrix for variable "p1lab",
    in equation "E_x1lab".

   Division by zero when this is not allowed.
   This occurred first when:

   numerator=0.0000000

   (There are 3 active indices)
   index "o" from set "OCC", value "skilled" (element 1)
   index "i" from set "IND", value "Dwellings" (element 20)
   index "XX01" from set "OCC", value "skilled" (element 1)

   Expression being evaluated is:
    (ALL,o,OCC) (ALL,i,IND) 0.3 * p1lab(i,o) +
        SUM(XX01,OCC,-0.3 *
     V1LAB(i,XX01) / V1LAB_O(i) * p1lab(i,XX01))

[Have written Coefficient values to headers "0001" to "0002"
 of file "og03ext2-assert-arith-fail.har".]

   This occurred doing  submatrix for variable "p1lab",
    in equation "E_x1lab".
```

In the above error report:

- the expressions for the numerator and denominator are not shown. [At present we are unable to show them when the error occurs in a Submatrix.] But the value of the numerator when the error occurred is shown (and, of course, the denominator was zero then).
- although the Variable **p1lab** occurs in the expression, its values are not written to the assert-arith-fail.har file. That is because, when calculating the entries of a Submatrix, the Variables have no values – rather their indices indicate what column in the Equations or LHS matrix the entries go in – see section 2.13 of GPD-1.

Example 4 – Integer overflow in a Formula

Consider the following Formula.[37]

```
Formula INVTOT = 131749*122101;
```

This leads to integer overflow since the product 131749*122101 is equal to 16,086,684,649. This is larger than the largest 4-byte integer 2,147,483,647 (a little more than 2000 million) which can be represented via the Fortran compilers used with GEMPACK. GEMPACK traps for integer overflow (see section 7.11). The arithmetic error report for this Formula is shown below.

```
%% Have encountered the following arithmetic problem(s)
  integer overflow

  This occurred doing  formula for coefficient "INVTOT",
   at line 6 in the TAB file.

   Expression being evaluated is:

     INVTOT = 131749 * 122101

   This occurred doing  formula for coefficient "INVTOT",
   at line 6 in the TAB file.
```

Note that there are no Coefficients (or Variables) in the relevant Formula so no arith-fail.har file is written.

Example 5 – Real overflow in a Formula

Consider the following TAB file, which is **ROFLO.TAB** in the standard examples supplied with GEMPACK.

```
Set COM (c1-c30) ;
Coefficient (all,c,COM) Coef1(c) ;
Coefficient (all,c,COM) Coef2(c) ;
Coefficient (all,c,COM) Coef3(c) ;
Formula (all,c,COM) Coef1(c)=$POS(c) ;
Formula (all,c,COM) Coef2(c)=$POS(c) + 4 ;
Formula (All,c,COM) Coef3(c)=[10.0^Coef1(c)]*[10.0^Coef2(c)];
Write Coef3 to terminal ;
```

When index c is about 20, Coef3(c) will be larger than 10^38 so real overflow occurs. The arithmetic error report (which you can see if you run TABLO on ROFLO.TAB and then use ROFLO.CMF) is shown below.

[37] This Formula was written by a GEMPACK user who reported a bug because Release 9 of GEMPACK reported a negative value for INVTOT. The negative value arose because of integer overflow which was not trapped for in Release 9 (but is trapped for in Release 10, as the error report in the text shows).

```
 %% Have encountered the following arithmetic problem(s)
    arithmetic overflow

   This occurred doing  formula for coefficient "Coef3",
    at line 7 in the TAB file.

    Expression being evaluated is:

      (ALL,c,COM) Coef3(c) =
          {[10.0  ]^[Coef1(c)]} * {[10.0  ]^[Coef2(c)]}

(Written real array, size 30, header '0001'.)
(Written real array, size 30, header '0002'.)
 [Have written Coefficient values to headers "0001" to "0002"
  of file "roflo-assert-arith-fail.har".]

    This occurred doing  formula for coefficient "Coef3",
    at line 7 in the TAB file.
```

Because this is arithmetic overflow, the report does not indicate the value of the index when the error first occurred, nor does it give details or values of the sub-expressions directly involved in causing the error. But the values of all Coefficients on the RHS are in the arith-fail.har file produced and you can use these values to track down the details of the error.

7.4.1 Updates – Fine Print

When a Coefficient is updated, the update occurs in two parts.

- Firstly any Update statements for this Coefficient are processed. These calculate the change in the Coefficient due to changes in the Variables during the current step.
- Secondly the new updated values for the Coefficient are calculated by adding the changes to the old value (that is, to the value before the current step).

Arithmetic errors of all 4 types (division by zero, invalid powers, invalid argument for a function, overflow) can occur during the first part. But only overflow can happen during the second part.

- If an arithmetic error occurs during the first part, the arithmetic error report will say that the error occurred **while updating Coefficient "<name>"**. The formula on the RHS of the relevant Update statement and its line number in the TAB file will be indicated in the report. If the error is one of the first 3 types, index values will be shown. The arith-fail Header Array file produced will contain the values of all Coefficients and Variables occurring on the RHS of the relevant Update statement.
- If an arithmetic error occurs during the second part, the arithmetic error report will say that the error occurred **while completing the update of Coefficient "<name>"**. No expression will be shown, nor any line on the TAB file indicated (since there is none relevant). The arith-fail Header Array file produced will contain the values of all Coefficients and Variables occurring on the RHS of ALL Update statements for the Coefficient.

Example

Suppose set COM has just 2 elements, c1 and c2. The following (rather unlikely) statements illustrate the points above.

```
Update (Change)  COEF1("c1") = SQRT[COEF2("c1")]*v1("c1") ;
Update (Change)  COEF1("c2") = v2("c2") ;
```

If an error occurs evaluating the RHS of either of these, the RHS expression will be shown and the relevant line number in the TAB file indicated. If the error is on the first statement, the arith-fail HA file will include the values of COEF2 and v1.

If an error occurs completing the update for COEF1, no RHS expression will be shown and the arith-fail HA file will include the values of COEF1 as well as Variables v1 and v2.

7.4.2 Other Arithmetic Errors

In the section above, we described reports of arithmetic errors which occur while doing a Formula, Submatrix, Backsolve or Update.

It is possible that arithmetic errors such as overflow or division by zero could occur in other places in a simulation. For example, overflow can occur while extrapolating the results or when combining the results from the current step with those from the previous steps. In such cases, you will not see a detailed error report like the ones in the previous section. However, in many cases the LOG file will contain details (such as, in the case of extrapolation, the component of the variable where the error occurred).

7.5 More Informative Creation Information

When a GEMPACK program creates a file, the program writes what we call **Creation Information** on the file. This Creation Information contains

- the time and date on which the file was created.
- the name and version of the program which created the file.
- the GEMPACK Release from which the program EXE was built.

This information is echoed (in the LOG file or to the terminal) when another GEMPACK program opens the file.

If you open a Header Array file in ViewHAR and then click on **History**, the top part of the form shown is the Creation Information for the file you are looking at.[38]

Example

Suppose that GEMPIE (see chapter 7 of GPD-4) is used to read the simulation results in the Solution file SJLB.SL4 produced by a Release 9 version of GEMSIM. Then GEMPIE echoes to the LOG file

```
!  This file was created at 15:51:15 on 08-MAY-2008 by the program
!    <GEMSIM  Version 3.6   January 2005>
!  which accesses some of the routines in the GEMPACK software release
!    <GEMPACK  Release 9.0-003  August 2006>
```

Or, if SJLB.SL4 was produced by a Release 9 version of the TABLO-generated program SJ.EXE, you would see

```
!  This file was created at 15:37:06 on 08-MAY-2008 by the program
!    <sj.for 08-MAY-2008 (a TABLO-generated program)>
!  which accesses some of the routines in the GEMPACK software release
!    <GEMPACK  Release 9.0-003  August 2006>
```

[38] However, you cannot see the Creation Information on a Solution file if you open the file with ViewSOL.

7.5.1 Fortran Compiler Added for HA Files (All Programs)

For Release 10, the name of the Fortran compiler used to build the executable image of the program which created the file is added at the end of the Creation Information of all Header Array files (but not for other files). For example

```
!  This file was created at 08:01:49 on 16-MAY-2008 by the program
!    <MODHAR   Version 6.0 - F90  February 1999>
!  which accesses some of the routines in the GEMPACK software release
!    <GEMPACK  Release 9.0 plus (final Rel 10 beta?)  May 2008>
!  [The program which created this file was compiled with LF95.]
```

Here we use the abbreviations[39]

- LF95 for Lahey/Fujitsu Fortran 95.
- LF90 for Lahey Fortran 90.
- IF32 for Intel Visual Fortran 32-bit version.
- IF64 for Intel Visual Fortran 64-bit version.

7.5.2 GEMSIM and TG-programs Add TAB and TABLO STI Names

Release 10 versions of GEMSIM and TABLO-generated programs usually add the name of the TAB file and (if one was used) of the STI file used to run TABLO to the Creation Information for all files (including Solution files and updated data files) they produce. The idea is that, when you see the names of the TAB and TABLO STI file used to produce the file, you will have a better idea of the provenance of the file and will better be able to analyse the information in it.

Example

Consider the condensation of Stylized Johansen produced using the Stored-input file SJCOND.STI supplied with the standard GEMPACK examples. Suppose that the standard SJLB simulation is run from this condensed version of SJ using a Command file SJCOND-LB.CMF (which is the same as SJLB.CMF except that it specifies the Auxiliary files as SJCOND). This will produce the Solution file SJCOND-LB.SL4.

Suppose that GEMPIE is used to read the simulation solutions in SJCOND-LB.SL4 produced by a Release 10 version of GEMSIM. Then GEMPIE echoes to the LOG file

```
!  This file was created at 16:00:46 on 08-MAY-2008 by the program
!    <GEMSIM v4.1 Jan 2008> [sj.tab,sjcondgs.sti]
!  which accesses some of the routines in the GEMPACK software release
!    <GEMPACK  Release 9.0 plus (almost final Rel 10 beta)  April 2008>
!  [The program which created this file was compiled with LF95.]
```

Or, if SJLB.SL4 was produced by a Release 10 version of the TABLO-generated program SJ.EXE, you would see

```
!  This file was created at 16:00:15 on 08-MAY-2008 by the program
!    <sjcond.for 08-MAY-2008> [sj.tab,sjcondtg.sti]
!  which accesses some of the routines in the GEMPACK software release
!    <GEMPACK  Release 9.0 plus (almost final Rel 10 beta)  April 2008>
!  [The program which created this file was compiled with LF95.]
```

Note the inclusion of the TAB and TABLO STI file names on the second line of the Creation Information in each case.

[39] These same abbreviations are also used to echo the Fortan compiler when a GEMPACK program starts running.

Fine Print

If long file names are used for the TABLO-generated program, the TAB file and/or the Stored-input file then there may not be enough space to hold all this information on a single line. When this happens, the TAB file name and Stored-input file name are abbreviated or dropped altogether. The string holding the program name and version is limited to 70 characters.

Strictly speaking, the Creation Information consists of 4 parts:

- the program name and version.[40] This goes on the second line when the Creation Information is echoed.
- the GEMPACK Release which was used to produce the executable image. This goes on the fourth line when the Creation Information is echoed.
- the time and date the file was created. This goes on the first line when the Creation Information is echoed.
- the abbreviation for the name of the Fortran compiler which was used to build the executable version of the program which created the file. This part is only echoed when the file is a Header Array file.

The change documented above changes the first of these in the sense that the names of the TAB and TABLO STI files used are added to the program name.

When writing this we noticed that we have not documented this Creation Information for earlier Releases of GEMPACK and apologise for that omission.

7.5.3 Header Array File History

Any Header Array file can contain what we call **History Information** (or simply **History**). This History consists of several lines of text (each line is limited to 60 characters) which are stored on the file and which are echoed when the file is read by another GEMPACK program. You can also see this History if you open the file in ViewHAR and click on the **History** menu item.

You can add to or change the History if you modify the file using MODHAR – see section 3.9.2 of GPD-4, or if you modify the file using ViewHAR.

The idea is that Creation Information (see section 7.5) and History help to remind you how, when and why you created the file. If you send the file to someone else, it should tell that person useful information about the file.

When you carry out a simulation, the updated versions of any Header Array data files have History written on them, as do the Solution file[41] and the Equations file (if produced). The History on these files is written by GEMSIM or the TABLO-generated program which creates them.

When writing section 7.5 above about Creation Information, we noticed that we had not properly documented History Information for earlier Releases of GEMPACK. We apologise for that omission.

[40] Each program has a version number and date. The version number is usually unrelated to the Release number. For example, in Release 9, GEMSIM was Version 3.6, January 2005 while in Release 10, GEMSIM is Version 4.1, January 2008. Similarly, in Release 9, TABLO is Version 5.5, February 2005 while in Release 10, TABLO is Version 6.1, January 2008.

[41] Solution and Equations files are Header Array files – see section 9.19 of GPD-5.

7.6 Accuracy Warnings from Iterative Refinement

In some cases, while doing iterative refinement (see section 12.1 of GPD-3), we are able to indicate that one or more results may be less accurate than you would like.

Iterative refinement is done at each step of a multi-step calculation as follows.

1. Solve the linearised equations after calculating the LU decomposition of the LHS Matrix.
2. Substitute the solution obtained so far into the original linearised equations to calculate the residuals (which are the differences between the original right-hand sides and those calculated by substituting the solution back into the equations).
3. Solve the linearised equations with the RHS replaced by the residuals to get corrections to the solutions obtained so far.
4. Produce new (hopefully more accurate) solutions by adding the corrections just obtained to the solutions previously obtained.
5. Repeat 2-4 above until the residuals are very small or until it seems not to be improving the solutions.

During iterative refinement you can see that we obtain several different candidates for the solution for each endogenous variable in the condensed system. We compare the largest of these (in absolute value) with the solution we use in accumulating (across all steps) the value we report (on the Solution file). This comparison may enable us to tell that the reported result for one or more variables is unlikely to have more than a very small number of accurate figures.

Example. In one ORANIGRD simulation[42] we found that the first candidate for x0imp("Dwellings") during step 1 of a 2-step Euler calculation was 1448833.2. The value taken (after iterative refinement) was –6.6232. So you can see that corrections indicated by iterative refinement were rather large. We normally hope to have up to 6 figures of accuracy. But because of the large correction involved here, we know that as many as 5 of these 6 figures have been lost, and hence that perhaps only 1 figure (the first "6") is accurate.

In this case the software will warn that the result for x0imp("Dwellings") is not very accurate. You will see the lines below in the LOG file.[43]

```
ITERATIVE REFINEMENT ACCURACY WARNINGS
--------------------------------------
%%WARNING. Relatively large changes during iterative refinement indicate
  that the Cumulative solution results for the following variables
  may not be very accurate. Please be careful when using/reporting
  results for these variables.
[NOTE. If you are able to backsolve for these variables,
        you may avoid these problems.]
 x0imp("Dwellings"). Result:-6.6232681, One candidate:1448833.2
  [Lose 5 figs.]
 [See section 7.6 of GPD-9.]
[End of iterative refinement accuracy warnings for Cumulative solution]
```

[42] This is the simulation in OGRD2A.CMF which is part of our TESTGP testing suite.

[43] The "One candidate" value shown is the largest (in absolute value) of the different candidates. The figures lost value is calculated using LOG10[BIGCAND/ABS(RESULT)] where BIGCAND is the "One candidate" value and RESULT is the value used from the current step in accumulating the result for this variable. A warning is given only if the above LOG10 is 3 or more (that is, 3 or more figures have been lost). Even if it seems that 3 or more figures have been lost, a warning is only given for results of "moderate" size – specifically if the result for the current step is at least 0.1 (%-change variable) or 5 (change variable).

7.7 New Additional and Target Shock-Type Statements

The "ashock" and tshock" statements are useful in Policy simulations with RunMONASH, RunDynam etc – see section 5.5.4 of GPD-3. These are "additional" and "target" versions of "shock" statements.

In Release 10, we have introduced similar "additional" and "target" versions of "final_level", "change" and "percent_change" statements. [These statements are described in sections 5.6 and 5.7 of GPD-3.]

We are grateful to James Giesecke for suggesting these statements.

Of course, as with "ashock" and "tshock" statements, these new statements can be included in any Command file. But they are really intended only for use in .PSH files used with RunMONASH etc.

Versions of RunMONASH etc from Version 3.29 (March 2008) allow these new keywords.

The new statements are

- "Additional" statements with key words **AChange** or **APercent_Change**. These are "additional" versions of "Change" and "Percent_Change" statements.[44]
- "Target" statements with key words **TFinal_Level**, **TChange** or **TPercent_Change**. These are "target" versions of "Final_Level", "Change" and "Percent_Change" statements.

These new statements are allowed in Command files for GEMSIM and TG-programs but not for SAGEM.

Examples

Suppose you have a levels variable LEV1 and you want to be sure that its post-simulation value in a RunMONASH Policy simulation will be 3.1. Then you can use the statement

```
TFinal_Level LEV1 = 3.1 ;
```

in your .PSH file. That will ensure the desired result even if LEV1 has a nonzero endogenous change in the Base simulation.

Suppose you have a Change linear variable Tariff_Rate and you want to be sure that its gets a shock of 10% over and above whatever happens to it in the Base Case simulation. Then you can use the statement

```
APercent_Change Tariff_Rate = 10 ;
```

in your .PSH file.

Note About AChange and APercent_Change Statements

APercent_Change is an additional %-change applied to the levels value calculated **after any ordinary shock has been applied**.[45]

AChange is an additional change applied to the levels value calculated **after any ordinary shock has been applied**.

Example 1

Suppose that you have a levels change variable LEV2 in your model and suppose that its pre-simulation value is 5. Consider the following statements

[44] Note that there is no "additional" version of "Final_Level" since we cannot see any natural meaning of such a statement.

[45] Here "ordinary shock statement" means "shock", "change", "percent_change" or "final_level" statements.

```
Shock LEV2 = 0.4 ;
APercent_Change LEV2 = 10 ;
```

The 0.4 shock to LEV2 is a change (since the associated linear variable is a change variable), so that increases LEV2 from 5 to 5.4. The APercent_Change is a %-change relating to the resulting value 5.4 (not to the pre-simulation value 5). Hence the effect of the APercent_Change statement is to increase LEV2 by 0.54 [10% of 5.4] from 5.4 to 5.94. Hence the effect of the two statements above is to increase LEV2 from 5 to 5.94. This is implemented via a shock (change) of 0.94 to LEV2. That is, the two statements above are the same as the statement

```
Shock LEV2 = 0.94 ; ! given that pre-sim LEV2 is 5
```

Example 2

Consider the following statements in a simulation based on the standard SJ.TAB starting from the standard SJ.DAT.

```
Shock p_XFAC("labor") = 10 ; ! The usual 10% increase in labor
AChange p_XFAC("labor") = 0.2 ; ! additional increase of 0.2 units
```

The pre-simulation value for XFAC("labor") is 4 (from SJ.DAT). So the first shock statement increases XFAC("labor") from 4 to 4.4. The AChange statement increases from 4.4 to 4.6. So the overall shock to p_XFAC("labor") is 15%. That is, the two statements above are the same as the statement

Shock p_XFAC("labor") = 15 ; !given that pre-sim XFAC("labor") is 4

Syntax Rules

- You cannot have two "additional" or two "target" statements applied to the same component of a variable. For example, the two "additional" statements

  ```
  ashock Tariff("C2") = 10 ;
  AChange Tariff = uniform 5 ;  ! error
  ```

  will produce an error.
- You cannot have an "additional" and a "target" statement both applying to the same component of a variable. For example, the statements

  ```
  Achange Tariff = uniform 5 ;
  TFinal_Level Tariff("c3") = 3.1 ;  ! error
  ```

  will produce an error.

See also section 7.17 for some clarifications about shock statements. These clarifications also apply to the statements documented above.

7.8 Condensation Information File

You can ask GEMSIM or a TABLO-generated program to write a **Condensation Information file** by including in your Command file a statement of the form

`condensation information file = <file-name> ;`

Here <file-name> should include the suffix and can include <cmf> – see section 7.16. For example

```
condensation information file = <cmf>-cond.txt ;
```

The Condensation Information file is a text file which contains information about the sizes and status (condensed, backsolved, substituted out, omitted) of the variables and equations. The sizes of run-time sets are known when this is done which makes this report different from the one written by TABLO on the Information file (when the sizes of run-time sets are not known). Two groups of variables are sorted in decreasing order of size:

1. variables in the condensed system
2. variables backsolved for, substituted out or omitted.

Similarly for equations.

You may find a Condensation Information file useful when you are deciding how to condense a new model or how to fine-tune the condensation for an old model. For example, substitute out or backsolve for the largest variables that are usually endogenous and perhaps omit some of the larger variables that are exogenous and unshocked in the current group of simulations.

7.9 New Functions for Log-Normal Distribution

Functions for the log-normal distribution are now allowed in TABLO Input files. We are grateful to Thomas Hertel for suggesting these.

A random variable X is said to have a **log-normal distribution** if its logarithm LOG(X) is normally distributed. See, for example, Wikipedia page

http://en.wikipedia.org/wiki/Log-normal

for information about the log-normal distribution.[46]

The new GEMPACK functions are

- **LOGNORMAL**. LOGNORMAL(X) is the probability density function for a positive variable X for which the natural logarithm LN(X) [that is, LOGE(X) in TABLO notation – see section 4.4.4 of GPD-2] is normally distributed with mean 0 and standard deviation 1. Note that

  LOGNORMAL(x) = [1.0/(x*SQRT(2*π))]*EXP(–LN(x)*LN(x)/2)

  where π is (approximately) 3.141592. Note also that LOGNORMAL(X) is only valid for values of X > 0.

- **CUMLOGNORMAL**. CUMLOGNORMAL(X) is the cumulative distribution function for a positive variable X for which the natural logarithm LN(X) is normally distributed with mean 0 and standard deviation 1. That is, CUMLOGNORMAL(X) is the probability that such a log-normally distributed variable is less than or equal to X (X>0). Note also that CUMLOGNORMAL(X) is only valid for values of X > 0.

- **GPERF**. GPERF(X) is the value of the so-called **Error Function** [usually denoted by ERF(X)]. GPERF(X) is equal to:

  $$\int_0^X 2*EXP(-T*T)/SQRT(\pi)\, dT$$

  where SQRT denotes square root. Note that GPERF(X) is valid for any X and that

  GPERF(–X) = –GPERF(X).

[46] It does not matter what base is used for the logarithm. If $LOG_B(X)$ is normally distributed for some base B, then $LOG_C(X)$ is also normally distributed for any other base C.

- **GPERFC**. GPERFC(X) is the value of the so-called **Complementary Error Function** [usually denoted by ERFC(X)]. GPERFC(X) is equal to:

  $$\int_0^{\infty} 2*EXP(-T*T)/SQRT(\pi)\ dT$$

  where SQRT denotes square root. Note also that for any X ,

  GPERFC(X) = 1 – GPERF(X).[47]

The Error Functions ERF and ERFC are useful for calculating values of the cumulative log-normal and cumulative normal functions – see, for example, the Wikipedia reference above, or section 6.2 of Press *et al* (1986). In particular,

CUMLOGNORMAL(X) = 0.5*[1 + GPERF(LN(X)/√2)]

and

CUMNORMAL(X) = 0.5*[1+ GPERF(X/√2)].[48]

A small tabulation of GPERF, GPERFC, LOGNORMAL, CUMLOGNORMAL is given below, together with GEMPACK's CUMNORMAL function.

x	–3	–2	–1	–0.5	0	0.001	0.5	1	2	3
GPERF(x)	–1.000	–0.995	–0.843	–0.520	0	0.001	0.520	0.843	0.995	1.000
GPERFC(x)	2.000	1.995	1.843	1.520	1	0.999	0.480	0.157	0.005	0.000
LOGNORMAL(x)	N.A.	N.A.	N.A.	N.A.	N.A.	0.000	0.627	0.399	0.157	0.073
CUMLOGNORMAL(x)	N.A.	N.A.	N.A.	N.A.	N.A.	0.000	0.244	0.500	0.756	0.864
CUMNORMAL(x)	0.001	0.023	0.159	0.309	0.5	0.500	0.691	0.841	0.977	0.999

The TABLO names of these functions LOGNORMAL, CUMLOGNORMAL, CUMNORMAL, GPERF and GPERFC are reserved words – see section 4.2 of GPD-2.

[47] GEMPACK code uses the approximation given in section 6.2 of Press *et al* (1986) to calculate GPERFC and then calculates GPERF(x) as equal to 1 – GPERFC(x). GEMPACK code uses GPERF to calculate the values for CUMLOGNORMAL and LOGNORMAL.

[48] For any real X, the value of CUMNORMAL(X) is equal to the integral of the standard normal curve from negative infinity to X (that is, to the area of the left tail from negative infinity up to X). The function CUMNORMAL is available in GEMPACK – see section 4.4.4 of GPD-2.

7.10 The RAS_MATRIX Function

A new function RAS_MATRIX has been added to TABLO syntax so that you can carry out a RAS procedure by writing a Formula in your TABLO Input file, for example:

```
Formula   ERROR_RAS =
RAS_MATRIX(InMat,RowTgt,ColTgt,OutMat,RowMult,ColMult,50);
```

This syntax is explained in section 7.10.4 below. The algorithm for the RAS procedure is adapted from the DAGG program written by Mark Horridge.

7.10.1 The RAS Procedure

In preparing input-output data, the need often arises to adjust a matrix so that it sums to given row and column totals. The RAS method meets this need.

Given an original matrix A(i,j), size r×c, and target vectors for the row totals R(i) and column totals C(j), the RAS procedure attempts to find a new, similar[49], matrix B(i,j) such that:

$\sum_j B(i,j) = R(i)$ $\quad$ i = 1,...,r (row constraint)

$\sum_i B(i,j) = C(j)$ $\quad$ j = 1,...,c (column constraint)

The new matrix B(i,j) is related to the original A(i,j) via:

$B(i,j) = rm(i)*cm(j)*A(i,j)$ $\quad$ i = 1,...,r j = 1,...,c

where rm(i) is a vector of row multipliers and cm(j) is a vector of column multipliers.

The RAS may be appropriately used to eliminate small inconsistencies that have arisen during data manipulation, or that may be traced to the use of data from several, mutually inconsistent, sources. Unfortunately, it will also smooth away gross errors due to blunders by the practitioner. Hence, it is vitally important to ensure that the RAS is performing small necessary adjustments to the data, rather than hiding the evidence of human carelessness. For this reason GEMPACK creates an extensive report of the RAS process, which goes to your log file as you run the TABLO-generated program. *This report should be carefully checked*.

7.10.2 An Intuitive Way to Think of the RAS

The RAS procedure is often explained and implemented in a simpler way. Again starting with an initial matrix and given row and column targets, the following procedures are repeated:

- Scale each row of the matrix so that it adds up to the corresponding row target.
- Scale each column of the matrix so that it adds up to the corresponding column target.

until the RAS "converges", ie, the scaled matrix adds to both row and column targets.

In favourable circumstances, the above procedure would produce the same scaled matrix as the GEMPACK procedure. Nevertheless, the GEMPACK procedure is preferable, because it provides, in the form of the row and column multipliers, an accurate account of how cruelly the original data had to be tortured in order to meet the targets. Remember, every RAS poses the modeller with **one** of these two questions:

- Why didn't my RAS converge?

[49] Entropy theory can be used to derive the RAS formulae, see: McDougall, Robert, 1999. "Entropy Theory and RAS are Friends", GTAP Working Paper 300, from *www.gtap.org*.

- My RAS converged, but have I done violence to the data?

GEMPACK provides sufficient diagnostics to answer either of these questions. Simpler routines, which can be easily implemented on a spreadsheet, usually provide insufficient diagnostics.

7.10.3 RAS Iterations

RAS_MATRIX follows an iterative procedure. Initially all elements of the row multipliers rm(i) and the column multipliers cm(j) are set to one. Then the following steps are repeated (perhaps many times).

(a) Using the current column multipliers, new row multipliers are chosen so that:

	$\sum_j B(i,j) = R(i)$	$i = 1,...,r$
i.e.:	$\sum_j rm(i)*cm(j)*A(i,j) = R(i)$	$i = 1,...,r$
i.e.:	$rm(i) = R(i) / \sum_j cm(j)*A(i,j)$	$i = 1,...,r$

(b) Using the current row multipliers, new column multipliers are chosen so that:

	$\sum_i B(i,j) = C(j)$	$j = 1,...,c$
i.e.:	$\sum_i rm(i)*cm(j)*A(i,j) = C(j)$	$j = 1,...,c$
i.e.:	$cm(j) = C(j) / \sum_i rm(i)*A(i,j)$	$j = 1,...,c$

Each of steps (a) and (b) disturbs the equality brought about by the previous step. However, successive adjustments to the row and column multipliers should become much smaller at every step. Usually, 5 or 10 iterations of (a) and (b) are sufficient to compute rm(i) and cm(j) (and thus B(i,j)) to a high degree of accuracy. The RAS "converges" when row and column constraints are satisfied, or when the multipliers stop changing.

RAS_MATRIX lets you choose how many iterations or steps are used. Each iteration is either a row scaling or a column scaling. The first and all odd-numbered iterations scale rows; even-numbered iterations scale columns.

If you specify an odd number of iterations, the final iteration will be odd (a row scaling), so you can expect that the scaled (output) matrix will add up exactly to the row targets -- but perhaps will not quite add up to the column targets. Conversely, if you specify an even number of iterations, column targets will be met exactly (but perhaps not row targets).

This feature could be useful. Suppose you wish to scale a MAKE supply matrix, size COM (commodity) * IND (industry), to meet SALES(COM) and COSTS(IND) targets. Assume that the scaled matrix did not exactly satisfy both targets. It might be that the row target SALES could be slightly adjusted (perhaps via inventory sales) but that the COST (column) target was fixed. In that case, you should request an even number of iterations, to ensure that the cost target was met exactly.

If you specify a large number of steps, the RAS may well converge: both constraints are satisfied, and so row and column multipliers stop changing. In this case, RAS_MATRIX will finish early, to save time. However, if you ask for an even number of iterations (say 100), you can be sure that the last iteration will be even (to favour column targets), even if only 40 iterations were needed. Similarly, if you ask for an odd number of iterations, RAS_MATRIX will finish with a row scaling.

Other uses for the multipliers

Apart from their diagnostic value, the multipliers have further uses. They can be used to scale groups of matrices. For example, we may wish to scale a basic values matrix and a tax matrix so that the sum of these two matrices—the producer price matrix—has given row and column totals. To do this, we would form the producer price matrix by adding the basic values and tax matrices together; then RAS the producer price matrix. The row and column multipliers would then be used to scale the basic values and tax matrices.

7.10.4 RAS Syntax in TABLO

The RAS_MATRIX function must be called as the right hand side of a Formula. There must not be any other terms on the right hand side of the formula. The syntax is:

```
Formula  <coefficient_name for real scalar> =
RAS_MATRIX(<coefficient_name for input matrix>,
     <coefficient_name for row target vector>,
     <coefficient_name for column target vector>,
     <coefficient_name for output matrix>,
     <coefficient_name for row multipliers vector>,
     <coefficient_name for column multipliers vector>,
     <integer_constant or coefficient_name for integer scalar>) ;
```

For example, the TABLO program in the text box below calls RAS_MATRIX by:

```
Formula  ERROR_RAS =
   RAS_MATRIX(InMat,RowTgt,ColTgt,OutMat,RowMult,ColMult,50) ;
```

Here the inputs are:

- InMat, dimensions Row*Col, the matrix to be scaled
- RowTgt, the vector of desired row sums (row targets)
- ColTgt, the vector of desired column sums (column targets)
- 50, the requested number of scalings

All the inputs must be given values (by READs or FORMULAs) before RAS_MATRIX is called. They will not be changed by the procedure.

The outputs will be:

- OutMat, the scaled matrix
- RowMult, the vector of row multipliers
- ColMult, the vector of column multipliers
- Error_RAS is the sum of the absolute values of the differences between RowTgt and the row sums of OutMat *plus* the summed absolute differences between ColTgt and the column sums of OutMat[50].

After calling RAS_MATRIX, these calculated values can be written to a file or used in other calculations.

The dimensions of RowTgt, ColTgt, OutMat, RowMult and ColMult have to match the dimensions of InMat.

Usually, RAS_MATRIX would be used in a TABLO program for data (ie, the TAB file would contain no equations or variables). In that case, each RAS_MATRIX formula would be executed just once. If you do use RAS_MATRIX within a model (ie, the TAB file contains equations and variables), you could use the (initial) qualifier to prevent the RAS being executed at every step of a multi-step simulation.

RAS_MATRIX is not allowed in a PostSim section of a TABLO program.

[50] Subject to the proviso that row targets and column targets have same sum. See point (b) in subsection 7.10.6.

Example TABLO-Input file for a RAS procedure

```
File   INFILE ;
 (new) OUTFILE ;
Set
 ROW read elements from file INFILE header "ROW" ;
 COL read elements from file INFILE header "COL" ;
Coefficient ! Inputs to RAS !
 (all,i,ROW)(All,j,COL) InMat(i,j) # Original matrix # ;
 (all,i,ROW)            RowTgt(i)  # Targets for Row totals # ;
 (all,j,COL)            ColTgt(j)  # Targets for Col totals # ;
Read
 InMat  from file INFILE header "ORIG" ;
 RowTgt from file INFILE header "RTGT" ;
 ColTgt from file INFILE header "CTGT" ;


Coefficient ! Output from RAS !
 (all,i,ROW)(All,j,COL) OutMat(i,j) # Matrix  after RAS # ;
 (All,i,ROW)            RowMult(i)  # Row multipliers # ;
 (All,j,COL)            ColMult(j)  # Column multipliers # ;
                        ERROR_RAS   # Error after RAS # ;
Formula
 ERROR_RAS = RAS_MATRIX(InMat,RowTgt,ColTgt,OutMat,RowMult,ColMult,50) ;

Write OutMat  to file OUTFILE header "SCAL" ;
Write RowMult to file OUTFILE header "RMLT" ;
Write ColMult to file OUTFILE header "CMLT" ;
Write ERROR_RAS to file OUTFILE header "ERAS";

ASSERTION  # RAS error small #  ERROR_RAS < 0.005 ;
```

7.10.5 The RAS Report

In the log file, a report is given about the progress of the RAS_MATRIX function. The report consists of five parts.

- **Task summary** giving the dimensions of the input matrix and the requested number of iterations.
- **Warnings** of potential problems: see below.
- **A table** comparing original row and column totals with the targets.
- **A listing** showing how (Target – Total) differences were reduced by each scaling, and by how much the multipliers were changing. As seen in the example log below, RAS_MATRIX finishes early if the multipliers stop changing.
- **A table** comparing final row and column totals with the targets, and showing the multipliers.

RAS Report Example

```
 ****  RAS REPORT  [see Section 7.10.5 of GPD-9] ****
 Matrix has    8 rows and    3 columns.
Requested number of iterations = 50
Even number of iterations, so last iteration will be a column scaling

        Sum of original Row targets was  47111.00000000000
        Sum of original Col targets was  47111.00000000000

%%WARNING 8: possible singleton problems at:
   Row  Col         Value      RowTotal      ColTotal
     6    1     6975.3696    13099.4541    10297.6074

 BEFORE RAS:
    Row    Row Total    Row Target    Target-Total  Target/Total
     1     2037.5520     2040.0000        2.4480      1.001201
     2     4745.4448     4745.0000       -0.4448      0.999906
     3     2132.1323     2132.0000       -0.1323      0.999938
     4    12132.8799    12131.0000       -1.8799      0.999845
     5     6013.1641     6024.0000       10.8359      1.001802
     6    13099.4541    13092.0000       -7.4541      0.999431
     7     3607.9265     3606.0000       -1.9265      0.999466
     8     3342.4443     3341.0000       -1.4443      0.999568
 Row Error (ie, sum of absolute differences) =        26.5659

    Col    Col Total    Col Target    Target-Total  Target/Total
     1    10297.6074    10141.0000     -156.6074      0.984792
     2    20750.6797    22065.0000     1314.3203      1.063339
     3    16062.7119    14905.0000    -1157.7119      0.927926
 Col Error (ie, sum of absolute differences) =      2628.6396
 Total RAS Error (ie, Row Error + Col Error) =      2655.2056

****  REPORT OF RAS ITERATIONS   ****
Iter   Type       Error     total multiplier change
   1   Row        26.5662        0.00484926
   2   Col      2638.6529        0.15096796
   3   Row       922.7787        0.17474401
   4   Col       694.7654        0.04632920
   5   Row       400.4696        0.07065874
   6   Col       282.7100        0.02014828
.............. (lines deleted)
  37   Row         0.0014        0.00000030
  38   Col         0.0013        0.00000006
  39   Row         0.0012        0.00000024
  40   Col         0.0008        0.00000000
  41   Row         0.0010        0.00000000
  42   Col         0.0008        0.00000000
RAS finished early; multipliers stopped changing.

 AFTER RAS:
    Row    Row Total    Row Target    Target-Total  Target/Total   Multiplier
     1     2040.0000     2040.0000        0.0000      1.000000      1.072251
     2     4745.0000     4745.0000        0.0000      1.000000      1.001884
     3     2132.0000     2132.0000        0.0000      1.000000      1.003800
     4    12131.0000    12131.0000        0.0000      1.000000      0.997987
     5     6024.0000     6024.0000        0.0000      1.000000      1.107376
     6    13092.0000    13092.0000        0.0000      1.000000      0.960344
     7     3605.9998     3606.0000        0.0002      1.000000      0.967073
     8     3341.0000     3341.0000        0.0000      1.000000      0.976912
 Row Error (ie, sum of absolute differences) =         0.0002

    Col    Col Total    Col Target    Target-Total  Target/Total   Multiplier
     1    10141.0000    10141.0000        0.0000      1.000000      1.018863
     2    22065.0000    22065.0000        0.0000      1.000000      1.076388
     3    14905.0000    14905.0000        0.0000      1.000000      0.890983
 Col Error (ie, sum of absolute differences) =         0.0000
 Total RAS Error (ie, Row Error + Col Error) =         0.0002

****  End of RAS REPORT   ****
```

7.10.6 Warnings of Potential Problems

RAS_MATRIX warns of potential problems that might prevent the RAS from converging. Such problems include:

(a) Some of the target row and column totals are vastly different from the row and column totals of the original matrix. To signal this problem, the RAS report prints out:

- the original row and column totals
- the target row and column totals
- the ratios of original row and column totals to the target—hopefully between 0.2 and 5.

There are warnings if any target or initial totals are zero or tiny.

(b) The sum of the target row totals is not equal to the sum of the target column totals. In this case, RAS_MATRIX scales one target vector to equalize the two sums[51].

(c) Negative elements in input matrix: these might prevent convergence, unless the negative element is small compared to other elements in the same row or column. The RAS report prints out a list of all negative elements of the original matrix which have absolute value greater than 5% of either of the corresponding row or column totals.

(d) Singletons—elements which are the sole non-zero entry in both their row and their column—force the RAS to scale a single number so that it equals both row and column targets. If the two targets differ, this will be impossible. Similarly, elements which are nearly as big as both their row and their column totals cause the RAS to perform poorly. To signal this problem the RAS report prints out a list of all elements in the original matrix which have absolute value greater than 50% of either of the corresponding row or column totals. There are two main remedies for the singleton problem:

(i) ensure target row and column totals corresponding to singletons are the same.

(ii) avoid singletons by 'smearing' the original matrix. This could be done by merely adding one to each element of the original matrix. For a more sophisticated approach, form the default apportionment matrix:

$$D(i,j) = R(i)*C(j)/T \qquad \text{where } T = \sum_j C(j) = \sum_i R(i)$$

D is the matrix that would arise from RASsing a matrix filled with ones. Replace the original matrix A with an average of A and D: e.g., $A(i,j) \Leftarrow 0.9*A(i,j) + 0.1*D(i,j)$

7.11 Handling Integers and Reals in GEMSIM and TG-Programs

Before Release 10, TABLO-generated programs behaved a little differently to GEMSIM when doing arithmetic, as explained below. From Release 10, these two programs behave in the same way.

The material in this section is rather technical. We suggest that most readers can happily ignore it.

Some problems prior to Release 10 which are now fixed are as follows.

1. MAX and MIN functions sometimes produced a TABLO-generated program which would not compile. For example, the statement

```
Formula Icoef1 = MAX(23, SUM(c,COM,Icoef2(c)) ;
```

[51] If you asked for an even number of iterations (to favour column targets), the row target would be adjusted if necessary. Actually, a local copy of the row target would be scaled – the input target is not changed.

where Icoef1 and Icoef2 are integer Coefficients, would result in Fortran code which would not compile with the LF90 and Intel Fortran compilers.[52]

2. Integer overflow was not detected. For example, the Formula

   ```
   Formula INVTOT =131749*122101 ;
   ```

   produced a NEGATIVE result for the non-integer Coefficient INVTOT because of undetected integer overflow when either GEMSIM or the TG-program was run.

3. SUMS sometimes produced different answers in GEMSIM and TG-programs. For example, the statements

   ```
   Set COM Size 20000000 ;  ! 20 million !
   Coefficient (Integer) SizeCOM ;
   Formula SizeCOM = SUM(c,COM, 1) ;
   ```

   produced the correct result 20000000 with GEMSIM but produced an incorrect result near to 16 million with a TG-program.[53] Now both GEMSIM and TG-programs produce the right answer.

4. Powers were handled differently depending on where they occurred. For example

   10^(-5)

   produced an arithmetic error if on the RHS of a Formula (but not inside a condition) for an Integer Coefficient, but otherwise correctly produced 0.00001. Now such expressions are treated the same wherever they appear. The expression 10^(-5) will now always lead to an arithmetic error (integer raised to a negative power). But 10.0^(-5) will always be interpreted as 0.00001.

7.11.1 Some TAB Files May Need Changes

Unfortunately the above changes made to the handling of integers and reals mean that a few GEMPACK users will need to alter TAB files which used to run ok. We apologise for this. Fortunately the changes are easy to make, as we explain below.

As explained in the next section, when an operation (except division) is made on two integers, integer arithmetic is used. That can lead to integer overflow or arithmetic problems which may surprise you. Often the "fix" is just to add ".0" after one of the integers, so that real (ie, non-integer) arithmetic is used.

These integer problems may show up when TABLO runs, or may not show until you run GEMSIM or the TABLO-generated program.

The examples below should make things clearer.

[52] The offending part of the code looked like "MAX(23, RS1)".

[53] The TG-program used real arithmetic to calculate the result. With single precision real arithmetic, it turns out that R1 + 1.0 = R1 when R1 is a single precision real greater than about 16 million.

Examples

```
Formula Coef1 = 10^10*Coef2 ;
```

This will lead to integer overflow since 10^10 is larger than the largest integer (2,147,483,647) allowed by GEMPACK – see section 7.11.3. The "fix" is to convert the first 10 to 10.0 so that the formula becomes

```
Formula Coef1 = 10.0^10*Coef2 ;
```

Then real arithmetic will be done when calculating 10.0^10. This problem will not show up until GEMSIM or the TG-program is run.

```
Formula Tiny = 10^(-8) ;
```

This will lead to an arithmetic error (integer raised to a negative power). Again just change 10 to "10.0" so that the formula becomes

```
Formula Tiny = 10.0^(-8) ;
```

and all will be well. This problem will not show up until GEMSIM or the TG-program is run.

```
Formula (all,c,COM) Coef1(c) = Coef2(c)/100000000000 ;
```

This will lead to an error when TABLO runs, since the integer in the denominator exceeds the largest allowed (2,147,483,647). Indeed TABLO will give you the hint:

```
    [If you want a real number, add ".0" at the end.]
```

So the fix is easy, just add ".0" at the end of the denominator so that the Formula reads

```
Formula (all,c,COM) Coef1(c) = Coef2(c)/100000000000.0 ;
```

See also section 7.4 for more information about the reporting of arithmetic errors.

7.11.2 Release 10 Details of Real and Integer Arithmetic

The material in this section is rather technical. We suggest that most readers can happily ignore it.

In Formulas, Updates and Equations, the programs distinguish between

- **real numbers**. The building blocks for these are non-integer Coefficients and real constants (strings of digits containing a decimal place, such as "23.41"), and
- **integers**. The building blocks for these are integer Coefficients and integer constants (strings of digits not containing a decimal point, such as "2461").

Certain functions (see section 4.4.4 of GPD-2) produce either a real or an integer as their output.

- The GEMPACK functions which produce an integer result are $POS, ROUND, TRUNC0 and TRUNCB.
- Output from the GEMPACK functions ABS, MAX and MIN depends on the type of the argument(s). ABS() produces the same type as its argument. MAX and MIN produce an integer output only when every argument is an integer – otherwise they produce a real result.
- All other GEMPACK functions (for example, EXP and LOG10) produce a real result, irrespective of the type(s) of the arguments.

When an arithmetic operation is carried out, the type of the result is as follows:

- If two integers are operated on, the result is an integer, unless the operation is division, in which case the result is a real.
- Otherwise the result is a real.

Example

If Icoef1 and Icoef2 are integer Coefficients and Rcoef1 and Rcoef2 are non-integer (ie, real) Coefficients, then

ICoef1*Icoef2 is an integer,
Rcoef1*Icoef2 is a real,

Icoef1/Icoef2 is a real,
234+Icoef1 is an integer,
234.1-Icoef2 is a real.

A SUM, PROD, MAXS or MINS operation produces output of the same type (real or integer) as the type of the expression being SUMmed or inside the PROD, MAXS or MINS.

Example

If Icoef1 and Icoef2 are integer Coefficients and Rcoef1 and Rcoef2 are non-integer (that is, a real) Coefficients, then

SUM(c,COM, ICOEF1(c)+23) produces an integer result.
PROD(c,COM, 1.1*ICOEF1(c)) produces a real result.

Prior to Release 10, all SUM, PROD, MAXS and MINS produced real output in TG-programs while GEMSIM followed the rules above. This unfortunate difference between the way GEMSIM and TG-programs produced incompatible results – see, for example, the SizeCOM example in section 7.11 above.

Powers A^B

The following are not allowed.

- Zero raised to the power zero.
- Zero raised to a negative power.
- Negative number raised to the power zero.
- Negative number raised to a non-integer power.
- Integer different from 1 and –1 raised to a negative integer power.

7.11.3 Integers and Reals Allowed

The largest integer allowed with GEMPACK is 2,147,483,647. This is the largest integer that may be stored in 4 bytes.

The largest real allowed with GEMPACK (before overflow occurs) is about 3.4*10^38. This is because GEMPACK programs use single precision reals (4 bytes).

7.12 GEMSIM Calculations

GEMSIM now does most of its intermediate calculations in double precision. In Release 9 and earlier, these calculations were done in single precision.

This will make arithmetic results from GEMSIM closer to those produced by the corresponding TABLO-generated program (see also section 7.11).

We are grateful to Kevin Hanslow for an example which prompted this change in GEMSIM.

We give some details in section 7.12.1 below for those who are technically inclined.

7.12.1 How GEMSIM and TG-Programs Do Arithmetic

Consider the simple formula

```
(All,c,COM) Coef1(c) = Coef2(c) + Coef3(c) + Coef4(c) ;
```

GEMSIM code uses 50 arrays RS1 to RS50 to hold the results of intermediate calculations. Only one operation is done at a time, with the results being stored in another RS array. For the formula above, the calculations go roughly as follows.

- Put the values of Coef2 into RS1 and the values of Coef3 into RS2.
- Do the first "+" by putting the result of the calculation RS1+RS2 into RS3.
- Put the values of Coef4 into RS4.
- Do the second "+" by putting the result of the calculation RS3+RS4 into RS5.
- Store the values of RS5 into Coeff1.

In GEMPACK Release 10, these RS arrays are declared as double precision reals. Previously they were declared as single precision reals.

For most calculations this makes little or no difference. However, if two large values of similar size are subtracted, there may be important differences. For example, in the formula above, if Coef2=6000000, Coef3=1 and Coef4= – 6000000 then

- in single precision, RS3 will equal 6000000 (since in single precision the system cannot distinguish between 6000000 and 6000001) so RS5 and hence Coef1 will equal zero.
- in double precision, RS3 will equal 6000001 and so Coef1 will equal 1, as expected.

The corresponding TABLO-generated program calculates this formula via the line of Fortran code:

```
C00001(c) = C00002(c) + C00003(c) + C00004(c)
```

Exactly how this is calculated depends on the particular Fortran compiler. But most compilers do this whole calculation using the double (or greater) precision built-in to the CPU. The conversion to single precision (when the results are put into Coef1) is only done at the end.

That is why using double precision arrays RS for the storage of intermediate calculations in GEMSIM gives results which are very close to those produced by TABLO-generated programs.

7.13 Tables and SSE Output from SLTOHT

If you separate two or more variables by colons ":" on a line of a Spreadsheet Mapping file, you can ask SLTOHT to produce tables of results in which the rows are the components of the variables and the columns are the different variables. This is described in detail in section 9.4 of the Release 8 version of GPD-4.

In that section of GPD-4, it is stated that such tables can only be produced if you have selected either option SS or SSS when running SLTOHT. In fact, such tables will also be output if you have selected option SSE "Spreadsheets with element labels".

The tables produced from several variables on the same line (separated by colons) are the same whichever of the three SS, SSS or SSE you have selected. But the output produced by other lines of your Spreadsheet Mapping file which contain a single variable name will be different. For example, if you put the GTAP variable **qo** on its own on one line of a Spreadsheet Mapping file, with option SSE you will see output like that shown in the qo(NSAV_COMM,REG) table of section 9.3 of GPD-4, whereas you will see output from SS or SSS which begins as shown below. [A different row is used for each commodity, region pair.]

```
(food:USA)       0.35
(food:EU)        0.26
```

You can produce tables of results corresponding to a single subtotal. To do so, respond 'u' (single subtotal) when prompted by SLTOHT and then indicate the subtotal number when prompted.

If you wish to specify the order of submatrices for arrays with 3 or more arguments using the %set notation described in section 9.3.2 of GPD-4, you cannot include other variable names (separated by colons) on the same line. As indicated in section 9.2.3 of GPD-4, the %set notation is only allowed when you have selected option SSE.

7.14 Assertion Failure Gives TAB File Line Number

If an assertion fails at runtime, the LOG file will now record the line number of the assertion within the TAB file. This is helpful if the TAB file contained several similar assertions.

7.15 Command File Required If Are Equations

Nearly always, you specify a simulation via a Command (CMF) file. But previous GEMPACK Releases allowed you to specify simulations interactively (by responding to command-line prompts). Starting with GEMPACK Release 10, that (old-fashioned) method is no longer available. We felt that it was not useful enough to justify the effort of maintenance and testing.

If there are Equations in the TAB file, we now require a Command file when you run GEMSIM or the TABLO-generated program.

For data-manipulation TAB files, we do not insist on a Command file. But we strongly advise you to use one.

7.16 Position of <CMF> in Command File Statements

As documented in sections 2.7 and 2.5 of GPD-3, <cmf> can be used in statements in Command files. It is replaced by the name of the Command file (with the suffix .cmf omitted).

For example, if you run a simulation by typing:

```
gemsim -cmf sim8.cmf
```

and sim8.cmf contained the line:

```
updated file TRQDATA = trq-<cmf>.upd;
```

GEMSIM would translate that line to become:

```
updated file TRQDATA = trq-sim8.upd;
```

In the past, `<cmf>` has sometimes[54] been replaced by the full path name (for example, c:\mysims\sim8.cmf), causing the `updated` line to contain an invalid file name:

```
updated file TRQDATA = trq-c:\mysims\sim8.upd;
```

which is treated as a fatal error. This anomaly has been fixed for GEMPACK Release 10; now `<cmf>` is replaced by *what appeared on the command line* [ie, the cmf name is **not** expanded to the full path name].

Following the above example, you would still have a problem if you typed:

```
gemsim -cmf c:\mysims\sim8.cmf
```

You could avoid such problems by editing the CMF line to read:

```
updated file TRQDATA = <cmf>-trq.upd ;
```

ie, placing `<cmf>` at the start of the filename. That would expand (acceptably) to:

```
updated file TRQDATA = c:\mysims\sim8-trq.upd;
```

Note that GEMPACK windows programs such as WinGEM and RunDynam launch simulations by executing command lines containing a full path-name. That is, they run commands like:

```
gemsim -cmf c:\mysims\sim8.cmf
```

[54] Particularly when GEMSIM, SAGEM or a TABLO-generated program was compiled using an Intel compiler.

To make this work, it is good practice to place `<cmf>` at the start of the filename.

7.17 Shock Statements – Some Clarifications

We have realised that section 5.5 of GPD-3, describing the syntax and semantics of Shock statements, did not say when the keywords "select from" are required after the "=" sign if shocks are being read from a file (see section 5.5.2 of GPD-3). We answer this question below.

When shocks are read from a file, the general form of the statement is

Shock <variable> <components> = file|select from file <filename> <header> ;

In this statement, <components> and <header> may be omitted.

As explained at the start of section 5.5 of GPD-3, if <components> is omitted, ALL EXOGENOUS components (which may or may not be all components) of the variable are shocked.

<components> may be specified via arguments as in

```
Shock p_XINTFAC("capital", IND) = select from file FACTOR.SHK ;
```

The following explains when "select from" is required.

- When you specify "select from", there must be exactly as many numbers on the file (or on the RHS) as there are components in <variable>. Then shocks to the specified components of <variable> are selected from the corresponding components in the file.
- When you do not specify "select from" the amount of data on the file (or on the RHS) must be exactly the same as the number of exogenous components of <variable>.

Examples

The examples below relate to variable f5(COM,SRC) from ORANIG01.TAB (supplied with the GEMPACK examples). In the normal aggregation as supplied in file ogozd867.har, set COM has 23 elements and SRC has the 2 elements "dom" and "imp", so variable f5 has 46 components. In the standard closure for ORANIG01, all 46 components of f5 are exogenous.

Example 1 Consider the statement

```
Shock f5(COM,"dom") = select from file shk1.har header "shk1";
```

This will be valid if there is a COMxSRC matrix of data at header "shk1" on file shk1.har. Just the "dom" part of that matrix will be used for the shocks.

Example 2 Consider the statement

```
Shock f5(COM,"dom") = file shk2.har header "shk2" ;
```

This will be valid if there is a COM matrix (that is, 23 numbers) at header "shk2" on file shk2.har. All 23 numbers there will be used for shocks in this case.

Example 3 Consider the statement

```
Shock f5("TCF", SRC) = select from  … ;
```

Here, the RHS ("...") should consist of 46 numbers -- from which 2 shocks will be selected.

Example 4 The statement

```
Shock f5("TCF", SRC) = 3.1 4.2 ;
```

means that f5("TCF","dom") is shocked by 3.1% while f5("TCF","imp") is shocked by 4.2%.

"Select from" cannot be used because variable f5 has 46 components, but only 2 numbers appear on the RHS.

Note that, if the amount of data on the RHS of the shock statement is the same as the number of components on the LHS, "select from" is redundant but can be used. So, in Example 2 above, "select from file" could be used.

The rules for when to use "select from" apply to all variants of shock statements including statements with key words "ashock" or "tshock" (see section 5.5.4 of GPD-3), "final_level", "change" or "percent_change" (see sections 5.6 and 5.7 of GPD-3) and the statements documented in section 7.7 above.

7.18 Complementarities - Speeding Up Single Euler Calculations

Erratum

Subtotals solutions are allowed during a complementarity simulation even there is no "accurate run", contrary to what is stated in section 16.5.6 of GPD-3.[55] The subtotals solutions are calculated during the approximate run, as is usually desirable (see chapter 7 of GPD-5).

Explanation

It is traditional to solve some large models, including the MONASH, MMRF and TERM models used at the Centre of Policy Studies at Monash University, using a single multi-step Euler calculation – for example, an Euler 8-step.[56]

When there are Complementarity statements in the model, the CPU time (see section 7.2) is roughly doubled compared to the same model with these Complementarity statements omitted. The doubling comes from the need to do the approximate run and then the accurate run.

When extrapolation is used, the accurate run usually produces a much more accurate solution than the approximate run. We strongly recommend that you do the accurate run when you would normally extrapolate from 2 or 3 multi-step calculations when solving your model.

But when the accurate run is done via a single Euler multi-step calculation, the accurate run results are probably hardly (if at all) more accurate than those from the approximate run – this is explained in section 7.18.1 below. In that case, it may make sense to omit the accurate run, since that will reduce the total CPU time by about 50%. The accurate run will be omitted if you include the statement

```
complementarity do_acc_run = no ;
```

in your Command file (see section 16.5.6 of GPD-3).

There are some restrictions as noted in section 16.5.6 of GPD-3, namely the solution method must be a single multi-step Euler, and only one subinterval is allowed and automatic accuracy is not allowed.

[55] This part of GPD-3 was written for Release 8.0 (October 2002). It has been possible (indeed desirable) to calculate subtotals on the approximate run (even when there are state changes) since Release 8.0-001 (October 2003). [We should have noted in GPD-5 that the prohibition on subtotals in section 16.5.6 of GPD-3 no longer applies.]

[56] Extrapolating from 2 or 3 multi-step calculations would usually (though not always) be possible. But for very large models, such calculations might take too long. For example, a 2-4-6 Gragg calculation would take as long as a 13-step Euler -- and might be less robust.

7.18.1 Accuracy of Approximate and Accurate Runs Using Euler

This compares the accuracy of the approximate and accurate runs when the accurate run is done using a single Euler multi-step calculation (eg Euler 8-step).

Recall that, during the approximate run, a step is normally[57] redone if there is a state change. The step is redone with a shorter step length so that the state change occurs just before the end of the step. This ensures that the simulation does not overshoot a bound by very much. See section 16.7.3 of GPD-3 for details.

This redoing of steps, coupled with the Newton correction terms (see, for example, Figure 16.1.2d in GPD-3), means that the approximate run should be nearly as accurate as the following accurate run if you are using just a single Euler multi-step calculation for the accurate run. But it may still result in small overshooting of bounds which should not happen during the accurate run.

Example

> Consider an import quota simulation such as in Figures 16.1.2a to 16.1.2d in GPD-3. There may be small overshooting of the quota bound on the approximate run (despite redoing steps and despite the Newton correction terms). But there should be no overshooting on the accurate run since then the closure is changed and a shock to the import volume is given so that it exactly hits the quota bound.

You may decide that the slight overshooting from the approximate run may be acceptable in order to gain the speed increase from not doing the accurate run.

7.19 TABLO Can Write a Linearised TAB File

If your TAB file contains levels equations, TABLO is able to write a linearised version of the file – that is, a version in which all levels equations are replaced by the corresponding linear equation.

If you ask AnalyseGE (Version 2.88 or later) to load a Solution file which was produced from a TAB file which contains levels equations, by default AnalyseGE loads the linearised TAB file rather than the original one. See also section 8.2.4.

We think that the linearised TAB file gives you more scope for analysing the Equations parts of your results.

We are grateful to Thomas Hertel for suggesting the possibility of loading a linearised TAB file into AnalyseGE.

7.19.1 Fine Print

If your TAB file contains levels equations, you can produce a linearised TAB file when you run TABLO interactively (as described in section 5.3 of GPD-1). Give option **LIN** (not shown on the screen) when TABLO starts running and then continue the run as normal. The linearised TAB file written by TABLO has "-lin" added at the end of the name. For example, if you process SJ.TAB, the linearised TAB file will be called SJ-LIN.TAB.

The linearised TAB file is a proper TAB file in the sense that it satisfies all TABLO syntax and semantic rules.

[57] Steps are not redone if you include the statement "complementarity redo_steps = no ;" in your Command file. We do not recommend this if you wish to get accurate results from the approximate run, as there may be considerable overshooting of bounds.

In the linearised TAB file, the levels equations of the original TAB file are shown as linearised equations. The form of the linearised equations is the same as is shown in the Information file produced when TABLO runs.

If there is a Formula & Equation in the original TAB file, the Formula remains in the linearised TAB file and it is followed by a linearised version of the associated levels equation.

The linearised TAB file would produce exactly the same results as you obtained from the original TAB file. That is because TABLO works with only with the linearised versions of the equations after the Check stage.

7.20 Unix Versions of GEMPACK

7.20.1 TABLO-generated Programs and Auxiliary Files on Unix

TABLO-generated programs now find their Auxiliary files on Unix in the same way that they do on Windows PCs (as described in section 3.2.1 of the Release 8 version of GPD-3). That is, TABLO-generated programs expect their Auxiliary files to be in the same directory and to have the same name (but with different extensions). This means that section 3.2.2 of the Release 8 version of GPD-3 no longer applies to Unix versions of GEMPACK.

Example

> Suppose that you are running the TABLO-generated program /home/mymodel/test/sj (derived from /home/mymodel/test/sj.f). The Auxiliary files must be in the same directory /home/mymodel/test and have the same "short name" sj. That is, the Auxiliary files must be /home/mymodel/temp/sj.axs and /home/mymodel/temp/sj.axs.

7.20.2 Unix Programs Set Exit Status

Unix versions of most GEMPACK programs now set the exit status when they finish. The status is set to zero if the program runs without error, or is set to 1 if the program ends with a fatal error.

This may be useful if you are running GEMPACK programs in a Unix script file to carry out batch jobs. [See section 7.2 of GPD-6 for the corresponding issue in DOS batch files.]

7.21 BCV Files No Longer Produced Or Supported

We have changed the internal organisation of GEMSIM and TABLO-generated programs so that BCV files (see chapter 9 of GPD-3) are no longer needed nor produced.

Consequences of this change are as follows.

- It is no longer possible to start from existing Equations and BCV files (see section 9.2 of GPD-3). However, it is possible to start from existing Equations and SLC files – see section 7.22.
- The old Command file statements

  ```
  BCV file = <file-name> ;      ! no longer allowed
  use BCV file <file-name> ;      ! no longer allowed
  ```

  are no longer allowed.

As part of this change, a temporary SLC file is created and used even when you have indicated that you do not want to produce an SLC file by putting the statement

`slc file = no ;` ! see section 8.4 of GPD-3

in your Command file.

7.21.1 System-Initiated Coefficients on SLC File

Because the SLC file is now read whenever the BCV file used to be read, the pre-sim values of all system-initiated Coefficients (things like C00843 introduced during condensation – see section 2.3.2 of GPD-2) are now always on the SLC file.

- Previously no such values were put on the SLC file unless the user had included the statement "SLC system_coefficients = yes ;" in the Command file (see section 8.4.2 of GPD-3).
- Now the statement "SLC system_coefficients = yes ;" only affects whether or not updated values of system-initiated Coefficients appear on the SLC file.

7.22 Starting from Existing Equations and SLC Files

It is now possible to start a simulation from existing Equations and SLC files. These files should both have been produced during the same run of GEMSIM or a TABLO-generated program since the values in them both reflect the pre-simulation values of the Coefficients in your model.

This is a replacement for the deprecated (and now no longer supported – see section 7.21) feature of starting a simulation from existing Equations and BCV files (see section 9.2.1 of GPD-3).

We expect to continue to support starting from existing Equations and SLC files into the future.

The SLC file contains pre-simulation values of all Coefficients (as well as information about the sets, subsets and set mappings). Hence the SLC file can be used to initialise the pre-simulation Coefficient part of the simulation in a way that is normally done during the preliminary pass and during the first step of the first multi-step calculation.

Now the statement

```
use Equations file <file-name> ;
```

is taken to mean that you want the program to start from the Equations file with that name and from the SLC file with the same name. If you want to specify a different SLC file name, you can do so via the statement

```
use SLC file <file-name2> ;
```

Examples

1. Suppose that you carry out a simulation with Command file SIM1.CMF which includes the statement

```
equations file = mymodel ;
```

but does not include a "Solution file = ;" statement. Then the Solution and SLC files produced will be called SIM1.SL4 and SIM1.SLC respectively and the Equations file MYMODEL.EQ4 will be saved.

Then suppose that you wish to carry out a second simulation SIM2.CMF starting from the same data as in SIM1.CMF and you wish to speed up this simulation by starting from the Equations and SLC files produced when SIM1.CMF ran. Then you should include the statements

```
use equations file mymodel ;
use SLC file sim1 ;   ! no SLC suffix required
```

in SIM2.CMF.

2. Suppose that all is as above except that SIM1.CMF includes the statement

```
equations file = <cmf> ;
```

Then the Equations file produced by SIM1.CMF will be called SIM1.EQ4 (and the SLC file produced will still be called SIM1.SLC). Then, in SIM2.CMF you just need to include the statement

```
use equations file sim1 ;
```

You do not need a separate "use slc file" statement since the SLC file you wish to use has the same root name (SIM1) as the Equations file.

When you start from existing Equations and SLC files,

- a new SLC file is produced. It takes its name from the name of the Solution file being produced so you need to make sure that name is different from the name of the SLC file you are starting from. If you have any postsim statements, the new SLC file also contains the postsim values of all Coefficients, so that part of the new SLC file will be different from the same part of the SLC file you are starting from (assuming you are giving different shocks etc).
- only the pre-simulation values of non-postsim Coefficients are read from the old SLC file. No postsim values are read from the SLC file.

7.22.1 Restrictions

The various restrictions that used to apply to starting from existing Equations and BCV files now apply to starting from existing Equations and SLC files. See section 9.2.1 of GPD-3.

In addition,

- it is not possible to start from Equations and SLC files if your model contains postsim sets. [This should have been the case for Release 9 when starting from Equations and BCV files but postsim sets were not checked by Release 9 software up to and including Release 9.0-003 (August 2006).]

CHAPTER 8

8 Changes to GEMPACK Windows Programs

8.1 Support for New Windows Versions

Since Release 9, GEMPACK has supported the standard 32-bit version of Windows XP. With Release 10 this is extended to 3 new versions of Windows:

- Windows Vista (32-bit)
- Windows Vista (64-bit)
- Windows XP (64-bit)

GEMPACK Release 10 has not been tested on earlier versions of Windows, and therefore these are not officially supported. We believe however that most programs will run fine on Windows 2000 or even Windows 98.

Windows Vista is evolving at the time of writing; we expect that new problems may emerge. See:

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

for a description of some Vista problems and work-arounds.

8.2 Changes to GEMPACK Windows Programs

A number of small changes have been applied to all the GEMPACK Windows programs:

- Buttons and other elements now follow the appearance you chose for your Windows XP or Vista "theme".
- New Help..GEMPACK PDF Documentation gives you direct access to the GEMPACK manuals. These now contain clickable links, to assist on-screen navigation.
- The online help file is now in the more modern CHM format (used to be HLP) and small changes to fonts and layout have been made to suit the default Vista fonts.

Each GEMPACK Windows program has a Help...What's New menu item which displays details of recent changes. Some of the more important changes to particular programs are indicated below.

8.2.1 Changes to WinGEM

- WinGEM will now run standard programs only if they are located in your GEMPACK Directory (previously it searched your Path).
- You can now customize the appearance of WinGEM (Options...Adjust Appearance).
- The WinGEM toolbar is no longer covered up if you maximize, say, ViewHAR.

8.2.2 Changes to ViewHAR

- You can now save real headers as a Database text file. This text format is used by some database programs. The same format can be very easily used to create an Excel Pivot Table.
- ViewHAR now behaves better if you have 2 monitors: window positioning is improved, and if you launch several ViewHARs they no longer appear right on top of one another. Similar improvements have been made to other GEMPACK windows programs.

- ViewHAR can now read and write GDX (GAMS) files, subject to various limitations.
- ViewHAR can now read and write HAR files in the new sparse storage format (see chapter 6).
- Edit..Create new header now allows you to directly specify the set associated with each matrix dimension (previously you had to do this later using Sets..Add or Modify Set Labels).

8.2.3 Changes to TABmate

- Mini-Gloss or Hints: Within a TAB file, if you hold down the Alt key (or the middle mouse button) and pass the mouse over any variable, coefficient or set, a hint appears showing the declaration (definition) of that item.
- File..Save All command saves all modified files.
- Tools...View/Create CMF command is used to build a template CMF for you to edit further.

8.2.4 Changes to AnalyseGE

- If the TAB file of your simulation contains levels equations, by default AnalyseGE loads a linearised version of that TAB file into the TABmate window instead of the original TAB file. The linearised TAB file (it contains linearised versions of the levels equations) gives you more scope for analysing the equations part of your model in AnalyseGE. The Options menu from the AnalyseGE window lets you choose to view the original TAB file.
  As an example, you might like to load the results from the standard 10% increase in labor supply with Stylized Johansen (SJLB.SL4) to see the differences. The standard TAB file SJ.TAB used to produce these results contains some levels and some linearised equations (see chapter 3 of GPD-1).
  See section 7.19 for more details about the linearised TAB file.
- A new, less irritating, closing dialog is used. And you have the option of saying that you never again wish to be asked about saving your results when you close AnalyseGE.
- AnalyseGE inherits various improvements to ViewHAR and TABmate (such as the Mini-Gloss feature mentioned above).
- New features of the TABLO language (eg, the RAS_MATRIX function) are handled correctly.

8.2.5 Changes to RunDynam, RunMONASH etc

- RunDynam, RunMONASH etc can run three jobs concurrently if your PC has three or more processors. [Prior to April 2006, the software would only allow 2 jobs to run concurrently.]
- Master/servant jobs can be run under RunDynam etc. See section 2.10 for details.
- New CMFSTART and Common rows have been added to the Closure/Shocks page.
- Up to 999 periods are allowed.
- The new additional and target shock-type statements (see section 7.7) are supported.

CHAPTER 9

9 New TABLO Syntax and Command File Statements

9.1 New TABLO Syntax

Functions LOGNORMAL, CUMLOGNORMAL, GPERF and GPERFC – see section 7.9.

9.2 New Command File Statements

Command (CMF) files for GEMSIM and TABLO-generated programs may now contain the following statements:

`WHS = YES|no ;` ! see section 6.1.1

`servants = NO|1|2 ;` ! see section 2.2

`condensation information file = <file-name> ;` ! see section 7.8

`use slc file <file-name> ;` ! see section 7.22

`pivots keep adding MA48 = yes|no ;` ! see section 5.2

`pivots modify MA48 = no|above|yes ;` ! see section 5.2

`AChange … ;` ! see section 7.7

`TChange … ;` ! see section 7.7

`APercent_Change … ;` ! see section 7.7

`TPercent_Change … ;` ! see section 7.7

`TFinal_Level… ;` ! see section 7.7

9.2.1 Changed Meaning

`SLC system_coefficients = yes|NO ;` ! see section 7.21.1

9.2.2 No Longer Allowed

`bcv file = <file-name> ;` ! see section 7.21

`use bcv file <file-name> ;` ! see section 7.21

10 REFERENCES

Press, W.H., B.P. Flannery, S.A. Teukolsky and W.T. Vetterling (1986), *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge.

11 GEMPACK DOCUMENTS

Harrison, W.J. and K.R. Pearson (2002), *An Introduction to GEMPACK*, GEMPACK Document No. 1 **[GPD-1],** Monash University, Clayton, Sixth edition, October 2002, pp.207+9.

Harrison, W.J. and K.R. Pearson (2002), *TABLO Reference*, GEMPACK Document No. 2 **[GPD-2]** Monash University, Clayton, Fourth edition, October 2002, pp.191+10.

Harrison, W.J. and K.R. Pearson (2002), *Simulation Reference: GEMSIM, TABLO-generated Programs and SAGEM*, GEMPACK Document No. 3 **[GPD-3]**, Monash University, Clayton, Second edition, October 2002, pp.262+12.

Harrison, W.J. and K.R. Pearson (2002), *Useful GEMPACK Programs*, GEMPACK Document No. 4 **[GPD-4]** Monash University, Clayton, Second edition, October 2002, pp.138+10.

Harrison, W.J. and K.R. Pearson (2005), *Release 9.0 of GEMPACK: New Features and Changes from Release 8.0*, GEMPACK Document No. 5 **[GPD-5]** Monash University, Clayton, First edition, April 2005.

Horridge, J.M., M. Jerie and K.R. Pearson (2008), *Installing and Using the Source-Code Version of GEMPACK on Windows PCs with Lahey or Intel Fortran*, GEMPACK Document No. 6 **[GPD-6],** Monash University, Clayton, Thirteenth edition, May 2008.

Horridge, J.M., M. Jerie and K.R. Pearson (2008), *Installing and Using the Executable-Image Version of GEMPACK on Windows PCs*, GEMPACK Document No. 7 **[GPD-7],** Monash University, Clayton, Tenth edition, May 2008.

Harrison, W.J. and K.R. Pearson (2002), *Getting Started with GEMPACK: Hands-on Examples*, GEMPACK Document No. 8 **[GPD-8],** Monash University, Clayton, Third edition, October 2002, pp.110+8.

Horridge, J.M., M. Jerie and K.R. Pearson (2008), *Release 10.0 of GEMPACK: New Features and Changes from Release 9.0*, GEMPACK Document No. 9 **[GPD-9],** Monash University, Clayton, First edition, May 2008.

12 INDEX