GEMPACK USER DOCUMENTATION
Release 9.0

GPD-5
Release 9 of GEMPACK:
New Features and Changes from Release 8.0

Release 9.0 of GEMPACK:

New Features and Changes from Release 8.0

GEMPACK Document No. GPD-5

W. J. Harrison

K. R. Pearson

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

First edition
April 2005

April 2007, reproduced in PDF with navigation links. Pagination may differ slightly from the April 2005 printing.

© Copyright 2005. The Impact Project and KPSOFT.

First edition
April 2005

ISSN 1030-2514

ISBN 0-7326-15054

This is part of the documentation of the GEMPACK Software System for solving large economic models, developed by the 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 9.0 (April 2005) of GEMPACK and the changes from Release 8.0 (October 2002).

The new features included are listed below.

- You can do post-simulation processing in the TABLO Input file for your model. You can now access in the post-sim parts of the TAB file pre-simulation and post-simulation values of Coefficients and the simulation results (values of the Variables – percentage changes or changes). You can write Formulas involving all of these. See chapter 2 for details.
- You can easily identify winners and losers from simulations. See chapter 3 for details.
- You can ask GEMSIM and TABLO-generated programs to use less memory if your model is taking too much memory. See chapter 4 for details.
- If the memory for nonzeros needs to be increased during LU decomposition, the software no longer needs to start the LU decomposition from the beginning again. See chapter 5 for details.
- For some models you may save considerable amounts of time by asking the programs to LU decompose the transpose of the Left-Hand Side Matrix or to use a different pivoting strategy. See chapter 5 for details.
- AnalyseGE can be used to analyse the results in a subtotal solution. You can decompose expressions you select with your mouse. See chapter 6 for details.
- You can calculate subtotals in simulations involving complementarities more reliably. See chapter 7 for details.
- There are many more minor enhancements. See chapters 8 to 11 for details.

Document Attributes

Name	Release 9.0 of GEMPACK: New Features and Changes from Release 8.0
Audience	CGE Modellers
Identifier	GPD-5

History :	Date	Author(s)	Comment
	April 2005	Jill Harrison and Ken Pearson	First Edition (Release 9.0)

[In the Release 5.1 (April 1994) documentation, GPD-5 was used for the GEMPACK document entitled "Installing and Using GEMPACK on Macintosh PCs with MacFortran/020" (May 1994).]

TABLE OF CONTENTS

1. RELEASE 9 OF GEMPACK – INTRODUCTION AND OVERVIEW 1-1

1.1 Contacting the Developers 1-1

1.2 Different Versions of GEMPACK and Associated Licences 1-1

1.3 Guide to the Documentation 1-1
1.3.1 For New GEMPACK Users – Getting Started 1-1
1.3.2 Guide For Experienced GEMPACK Users 1-2

1.4 New Features in Release 9.0 1-2

1.5 The GEMPACK Programs 1-2

1.6 Models Supplied with GEMPACK 1-2

1.7 GEMPACK World-Wide Web Site 1-3

1.8 GEMPACK-L Mailing List 1-3

1.9 Machines Supported 1-4

1.10 Beta Testers 1-4

2. POST-SIMULATION PROCESSING IN YOUR MODEL'S TAB FILE 2-5

2.1 Examples – Summaries of the Updated Data 2-6
2.1.1 Adding Some Post-sim Processing to ORANIG03.TAB 2-6
2.1.2 Adding Some Post-sim Processing to GTAP61.TAB 2-8
2.1.3 New File Can Contain Both the Original and Updated Data 2-9

2.2 Examples – Post-Simulation Processing of Variable Results 2-10
2.2.1 GTAP – Table of Important Regional Results 2-10
2.2.2 TERM – Table of Important Regional Results 2-11
2.2.3 ORANIG03 – Table of Winning and Losing Industries 2-12
2.2.4 ORANIG03 - Tariff cut Simulation Results 2-14

2.3 Sophisticated Post-Simulation Processing 2-15
2.3.1 GTAP – Sophisticated Post-Simulation Processing in DECOMP.TAB 2-15
2.3.2 Other Models 2-18

2.4 Advantages of Post-Sim Processing in TAB File – Overview 2-19

2.5 New TAB File Statements, Syntax and Semantic Rules 2-20
2.5.1 Statements Allowed in Post-Simulation Sections of a TAB File 2-20
2.5.2 Syntax and Semantics for Formulas in Postsim Part of TAB File 2-21
2.5.3 Semantics for Reads in Postsim Part of TAB File 2-22
2.5.4 Semantics for Assertions in Postsim Part of TAB File 2-22
2.5.5 Semantics for ZeroDivides in Postsim Part of TAB File 2-23
2.5.6 Default Statements and Certain Qualifiers Not Relevant in Postsim Part 2-23
2.5.7 Post-Sim Sets, Subsets, Coefficients not Allowed in the Ordinary Part 2-24
2.5.8 PostSim Qualifier for Write or Display 2-24
2.5.9 Implications for Condensation 2-24

2.6 Postsim Extra Statements in a Command File 2-25

2.7 Suppressing Post-Simulation Processing 2-25

2.8 When and How Is Post-Simulation Part Done? 2-26
2.8.1 Example – SJPS.TAB and SJPSLB.CMF 2-27
2.8.2 Other Examples 2-28

2.9 Postsim Parts of TAB File and AnalyseGE 2-29
2.9.1 Coefficients or Expressions Selected from the TABmate Form 2-29
2.9.2 Expressions Evaluated from the AnalyseGE Form 2-30
2.9.3 UDC Files Now Redundant? 2-30

2.10 Postsim Sets and Subsets not on Equations File 2-30

3. RANKING SETS VIA DATA (FINDING WINNERS AND LOSERS) 3-32

3.1 Identifying Winners and/or Losers 3-32
3.1.1 Example OG01PS.TAB 3-33

3.2 Ranked Set Statements – Syntax and Semantics 3-35

3.3 Converting Non-Intertemporal Sets to Intertemporal Sets and Vice Versa 3-36
3.3.1 TABLO Statements 3-37
3.3.2 Household Income Example in Detail 3-37

4. WHEN MEMORY IS SHORT (GEMSIM OR TG-PROGRAMS) 4-40

4.1 Memory Reports 4-40

4.2 Memory Sharing 4-41
4.2.1 When Memory Sharing May Be Useful 4-41
4.2.2 Running in Virtual Memory is Usually Very Slow 4-41
4.2.3 Restrictions 4-42
4.2.4 What Arrays Share Memory 4-42
4.2.5 Main Memory Saving Comes During LU Decomposition 4-42
4.2.6 Memory Saving If Using MA48 and MMNZ2 is Smaller than MMNZ 4-43
4.2.7 CPU Time Penalty 4-43
4.2.8 Large Work Files Are Needed 4-43
4.2.9 GTAP Examples 4-44
4.2.10 USAGE Examples 4-45
4.2.11 Less Total Memory is Used Than in Release 8.0 4-45
4.2.12 Technical Notes About Memory Sharing 4-46
4.2.13 Technical Fine Print about TABLO-generated Programs 4-46

5. LU DECOMPOSITION 5-48

5.1 Memory for LU Decomposition in Simulations 5-48
5.1.1 Alerting About MMNZ, MMNZ1, MMNZ2 Re-Allocations 5-49

5.2 Memory Reallocation Now Done Within the MA48 Routines 5-49
5.2.1 Compressions if Using New MA48A Routine 5-50
5.2.2 Reallocations if Using the New MA48 Routines 5-51
5.2.3 Reallocations if Using the MA28 Routines 5-52

5.3 LU Decomposing Transpose of LHS May be Much Quicker 5-52
5.3.1 Timings with Different Models 5-52

5.4 Alternative Pivot Strategy in MA48 5-53

6. ANALYSEGE 6-56

6.1 Subtotals 6-56
6.1.1 SLTOHT Can Select a Subtotal 6-56

6.2 Decomposing Selected Expression 6-56

7. SUBTOTALS WITH COMPLEMENTARITY STATEMENTS 7-58

7.1 Telling the Software When to Calculate Subtotals 7-59

7.2 Calculating Subtotals on the Approximate Run - Recommended 7-60
7.2.1 Downside – Decomposition Results do not Add Exactly to Accurate Cumulative Solution 7-60

7.3 Calculating Subtotals on the Accurate Run – Not Recommended 7-61
7.3.1 Decomposing Accurate – Include Extra Variables in Subtotals 7-61

7.4 Examples: MOIQ3.CMF and Related Simulations 7-62
7.4.1 Example: MOIQ3AC.CMF Simulation 7-65
7.4.2 Several Subintervals 7-65
7.4.3 MOIQ3.CMF Subtotals Calculated During Approximate Run 7-66
7.4.4 MOIQ3AC.CMF Subtotals Calculated During Accurate Run (Not Recommended) 7-67
7.4.5 Subtotals from Accurate Run Not Robust 7-68
7.4.6 Subtotals from Approximate Run Are Robust 7-70
7.4.7 The Initial Closure is Shown on Solution Files 7-70

7.5 Variables with No Components Exogenous Allowed in Subtotals 7-70

8. ENHANCEMENTS TO THE TABLO LANGUAGE 8-72

8.1 Reads Only if Header Exists – Read Qualifier IfHeaderExists 8-72
8.1.1 Syntax and Semantic Rules 8-72

8.2 Default Statements for Bounds on Coefficients in TAB Files 8-73

8.3 Reserved Words Can Be Used As Set Element Names in TAB Files 8-73

8.4 BY_ELEMENTS Qualifier and Set Mappings 8-74
8.4.1 BY_ELEMENTS Qualifier in Formulas for Set Mappings 8-74
8.4.2 Writing Set Mappings BY_ELEMENTS 8-74
8.4.3 Updating Set Mappings Read BY_ELEMENTS 8-74

8.5 Seeing the Values of Integer Coefficients of Dimension 3 or Higher 8-75

8.6 Names for Homotopy Variables Added to Levels Equations 8-75

8.7 Clarifications Relating to Release 8 Documentation 8-77
8.7.1 Set Statements 8-77
8.7.2 Formula Statements 8-77
8.7.3 Write Statements 8-78
8.7.4 Equation Statements 8-78

9. ENHANCEMENTS TO SIMULATION PROGRAMS 9-80

9.1 SSE Output 9-80
9.1.1 SSE Output from GEMSIM and TABLO-generated Programs 9-80

9.1.2 SSE Output has Coefficient in Top LHS Corner 9-80

9.2 Topics Relating to Simulation Accuracy (GEMSIM and TG-programs) 9-81
9.2.1 Smiling or Frowning Face Numbers are Reported 9-81
9.2.2 Fatal Error if Accuracy Too Low 9-81

9.3 Levels Results for Change Linear Variables 9-82
9.3.1 GEMPIE and SLTOHT 9-82
9.3.2 ViewSOL 9-82
9.3.3 RunGEM 9-83

9.4 Closure Specified Using Data Files 9-83

9.5 SLC, UDC and AVC Files 9-84
9.5.1 SLC Files May Contain Updated Values 9-84
9.5.2 UDC and AVC Files Written on a Postsim Pass 9-84
9.5.3 SLC, UDC and AVC Files Contain Values of Postsim Coefficients 9-84

9.6 Checking for Duplicate Elements in a Set 9-85

9.7 When an Assertion Fails 9-85

9.8 Assertions, Writes on Preliminary Pass and Postsim Pass 2 9-86
9.8.1 Assertions and Writes to Terminal May Be Done on Preliminary Pass 9-86
9.8.2 Assertions and Writes to Terminal May Be Done on Postsim Pass 2 9-86

9.9 Set and Element Names on Extrapolation Accuracy (XAC) Files 9-88

9.10 Changes to How Extrapolation is Done 9-89
9.10.1 Shocked Variables 9-89

9.11 Accuracy Summary Information Now on Solution File 9-90

9.12 Swap Statements on Command Files 9-90

9.13 Command File Statements for Newton's Method 9-91
9.13.1 Details of the Newton's Method Command file Statements 9-92
9.13.2 Example: SJLVLBNW.CMF 9-93
9.13.3 Example: SJLVLBNW2.CMF 9-94

9.14 Reporting Newton Error Terms in Models with Levels Equations 9-95
9.14.1 Example: Reporting Newton Errors with SJLVLBER.CMF 9-95
9.14.2 Command File Statements for Reporting Newton Error Terms 9-96

9.15 Suppressing Arithmetic Errors in GEMSIM and TABLO-generated Programs 9-98

9.16 Rate% Shock Statement 9-98
9.16.1 Fine Print about Rate% Shock Statements 9-100

9.17 Shock Statements using a Slice from a Header Array file 9-101
9.17.1 Use with RunDynam, RunMONASH and related Windows programs 9-102
9.17.2 Syntax Rules for Slice Shock Statements 9-103

9.18 SSL and Subtotals 9-103

9.19 Solution, Equations and Environment Files are Header Array Files 9-103

9.20 Several Subintervals – Clarification 9-104

10. ENHANCEMENTS TO UTILITY PROGRAMS 10-106

10.1 New Command Line Options for Some Programs 10-106
10.1.1 Options –log, –lon and –los 10-106
10.1.2 TAB file and WFP/PGS on Command Line with TABLO 10-106
10.1.3 SLTOHT – Solution File on Command Line 10-107
10.1.4 CMPHAR and Files on the Command Line 10-107

10.2 Deleting Temporary Lahey/Fujitsu Copies of Binary Files 10-108

10.3 Suppressing Arithmetic Errors in ACCUM, DEVIA and CMBHAR 10-108

10.4 SLTOHT 10-108

10.5 CMPHAR Reports Difference Metric Values 10-109

10.6 SEEHAR Option FNP 10-109

11. GEMPACK WINDOWS PROGRAMS 11-110

11.1 MIN Files Not Needed by RunGEM for Release 9 Programs 11-110

11.2 ViewSOL and Release 9 Solution files 11-110

11.3 AnalyseGE 11-110

11.4 Enhancements to Other Windows Programs 11-110

11.5 RunMONASH, RunDynam and RunGTAP 11-110

12. RELEASE 9 OF GEMPACK WITH RUNMONASH, RUNGTAP 12-112

12.1 RunMONASH, RunDynam etc 12-112

12.2 RunGTAP 12-112

13. NEW RELEASE 9 TABLO STATEMENTS AND QUALIFIERS 13-114

14. NEW RELEASE 9 COMMAND FILE STATEMENTS 14-116

14.1 GEMSIM, TABLO-generated Programs and SAGEM 14-116

14.2 Just GEMSIM and TABLO-generated Programs 14-116

15. REFERENCES 15-118

16. GEMPACK DOCUMENTS 16-118

17. INDEX 17-120

CHAPTER 1

1. Release 9 of GEMPACK – Introduction and Overview

This document describes the new features in Release 9.0 (April 2005) of GEMPACK and the changes from Release 8.0 (October 2002).

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. The only new documents for Release 9 are this document GPD-5 and the installation documents GPD-6 and GPD-7.

1.1 Contacting the Developers

For more information about GEMPACK, contact

Professor Ken Pearson
Centre of Policy Studies and Impact Project
Monash University, (Box 11E)
Clayton, Vic 3800
Australia
email: Ken.Pearson@buseco.monash.edu.au
telephone: (03)-9905-5484 or +61-3-9905-5484 from overseas
(03)-9905-5112 or +61-3-9905-5112 (CoPS admin staff)
FAX: (03)-9905-2426 or +61-3-9905-2426 from overseas

1.2 Different Versions of GEMPACK and Associated Licences

There are the same versions and licences for Release 9.0 as are described in section 1.9 of GPD-1.

However new licence files are required for use with Release 9.0 of GEMPACK.

1.3 Guide to the Documentation

We provide two alternative guides. Readers new to GEMPACK should read the one in section 1.3.1 while readers familiar with Release 8.0 (or a previous version) of GEMPACK should read the one in section 1.3.2.

This document contains **an index** which should make it easier for you to find the information you may need at different times in the future. 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

The document GPD-1 is the starting place if you are a new GEMPACK user. Follow the guide in section 1.5.1 of GPD-1.

You should you not begin reading this document GPD-5 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-5 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 8.0 (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 9.0 are listed in section 1.4 below. If you have not used Release 8.0, you will find a list of the new features introduced in Release 8.0 in section 1.6 of GPD-1 and the previous releases in chapter 7 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 9.0

The new features included in Release 9.0 are listed below.

- You can do post-simulation processing in the TAB file for your model. You can now access in post-sim parts of the TAB file pre-simulation and post-simulation values of Coefficients and the simulation results (values of the Variables – percentage changes or changes). You can write Formulas involving all of these. See chapter 2 for details.
- You can easily identify winners and losers from simulations. See chapter 3 for details.
- You can ask GEMSIM and TABLO-generated programs to use less memory if your model is taking too much memory. See chapter 4 for details.
- If the memory for nonzeros needs to be increased during LU decomposition, the software no longer needs to start the LU decomposition from the beginning again. See chapter 5 for details.
- For some models you may save considerable amounts of time by asking the programs to LU decompose the transpose of the Left-Hand Side Matrix or to use a different pivoting strategy. See chapter 5 for details.
- AnalyseGE can be used to analyse the results in a subtotal solution. You can decompose expressions you select with your mouse. See chapter 6 for details.
- You can calculate subtotals in simulations involving complementarities more reliably. See chapter 7 for details.
- There are many minor enhancements. See chapters 8 and 11 for details.

If you use RunGTAP or RunDynam or RunMONASH etc, there are some simple changes you will need to make so that these programs can handle models implemented with Release 9.0 of GEMPACK – see chapter 12 for details.

Summaries of the new TABLO and Command file statements for Release 9.0 are given in chapters 13 and 14 respectively.

If you use Source-code GEMPACK on a Windows PC, there are some changes you may wish to be aware of. See the Release 9 version of GPD-6.

1.5 The GEMPACK Programs

These are as listed in section 1.7 of GPD-1.

1.6 Models Supplied with GEMPACK

All the models described in section 1.8 of GPD-1 are supplied with Release 9.0 of GEMPACK.

In addition, some extra ones, as detailed below, are supplied.

ORANIG03. This is the 2003 version of the ORANI-G model.

TERM. This is a regional model of a single country (such as Australia or China) developed by Mark Horridge and colleagues at the Centre of Policy Studies, Monash University. See

http://www.monash.edu.au/policy/term.htm
for more details.

We also provide several TAB files with post-simulation sections added. You will use these when you carry out the hands-on examples described in chapters 2 and 3. These TAB files (each of which has associated Command files) include the following.

- **OG01PS.TAB**, an extension of ORANIG01.TAB. See section 3.1.1.
- **SJPS.TAB**, an extension of SJ.TAB. See section 2.8.1.
- **GT61PS.TAB** and **GT61P2.TAB**, both extensions of GTAP61.TAB. See sections 2.1.2 and 2.1.3.
- **OG03PS.TAB**, an extension of ORANIG03.TAB. See section 2.1.1.
- **TERMPS.TAB**, an extension of TERM.TAB. See section 2.2.2.

All of the hands-on examples referred to in this document GPD-5 can be run with either a Source-Code or Limited Executable-Image Version of GEMPACK.

1.7 GEMPACK World-Wide Web Site

The GEMPACK World-Wide Web site can be accessed at address:

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

This 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 Ken Pearson.

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).

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. Information about subscribing and sending messages is below.

1. **To subscribe to GEMPACK-L**, send the one-line mail message (as the text of the mail message)

   subscribe gempack-l

   to the address **mailserv@cc.monash.edu.au**
   [Don't send this message to gempack-l@vaxc.cc.monash.edu.au since then your request will go to all readers on this list, but not to the program which adds you to the list.]

2. **To send mail** to all other subscribers on the list, send your mail to the address

   gempack-l@vaxc.cc.monash.edu.au

3. To read mail sent to GEMPACK-L
   You should be able to use your usual mail utility to read GEMPACK-L mail and to send messages to it. [The address to which GEMPACK-L mail will be sent is the address from which the mail message "subscribe gempack-l" was sent.]
4. **To leave the list** at any time, send the one-line message

   unsubscribe gempack-l

   to the MAILSERV address **mailserv@cc.monash.edu.au**
   [Don't send this message to gempack-l@vaxc.cc.monash.edu.au since then your request will go to all readers on this list, but not to the program which adds you to the list.]

1.9 Machines Supported

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

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

We are grateful to Joe Francois for suggesting that we support Macintosh PCs and for testing and documenting the installation of Release 9 of GEMPACK on Macintosh PCs running version 10.3 or later of OS X.

1.10 Beta Testers

We are grateful to Martina Brockmeier, Joaquim Bento S. Ferreira Filho, Kevin Hanslow, Tom Hertel, Lars-Bo Jacobsen, Jong-Hwan Ko, Frank van Tongeren, Tien Pham Duc, Marinos Tsigas, several of our colleagues at the Centre of Policy Studies and any others who have been beta testers for Release 9.

CHAPTER 2

2. Post-Simulation Processing in your Model's TAB file

This is the major new feature in Release 9 of GEMPACK.

The idea is that you can add statements to the TAB file of your model to carry out calculations **post-simulation**. This means that you can carry out calculations which depend on the simulation results, and you can create tables and write the results to text or Header Array files post-simulation.

The calculations you perform and the tables you create can be based on

- the pre-simulation values of Coefficients,
- the post-simulation values of Coefficients, and
- the simulation results (values of the Variables – percentage changes or changes).

These three kinds of values are available and can be processed using **Formulas** and **Write** (or Display) statements in the post-simulation part of the TAB file. These post-simulation Formulas and Write statements are carried out after the simulation has been completed and the results and updated data written in the usual way. In particular, you can access simulation results in this part of the TAB file as if the Variables were Coefficients.

What do we mean by the **post-simulation** or **updated values** of a Coefficient?

> You can think of the post-simulation or updated values as the values of Coefficients after all the shocks have been applied in the simulation. If the Coefficient is read in and updated, then the post-simulation values are of course the ones written to the relevant updated data file after the simulation. If the Coefficient is not read in, but is calculated from a Formula, the post-simulation values of the Coefficient are calculated from the same Formula using updated or post-simulation values of the Coefficients on the right-hand side of the Formula.

In this chapter

- we give several examples of post-sim processing in a TAB file (see sections 2.1 and 2.2). These are all based on Examples files distributed with Release 9 of GEMPACK. We encourage you to work through these examples on your computer as you are reading about them.
- we suggest that sophisticated post-simulation processing (such as is currently available with the GTAP model and some other standard models) could more easily now be done in the main TAB file for the model. We also suggest that such sophisticated post-simulation processing can now easily be added to all models. See section 2.3 for details.
- we summarise in section 2.4 what we see as the advantages of using post-sim processing in your TAB files.
- we set out the formal syntax and semantic rules for post-sim processing parts of TAB files in section 2.5.
- we tell in section 2.6 you how you can add post-sim processing TABLO-like statements to your Command files.
- we describe how and when this post-simulation processing is carried out. See section 2.8 for details.
- we describe how AnalyseGE gives easy access to both pre-simulation and post-simulation values whenever you open a Solution file produced from a TAB file which contains post-sim sections of code. See section 2.9 for details.

There is a lot here. We suggest that you work through this chapter slowly, doing all of the hands-on examples as you go.

2.1 Examples – Summaries of the Updated Data

Many models write summaries of the salient features of a database as part of the TAB file used to solve the model, or in a related TAB file. Below you can work through examples based on ORANIG03.TAB and GTAP61.TAB. The relevant TAB and Command files are in the GEMPACK examples supplied with Release 9.

2.1.1 Adding Some Post-sim Processing to ORANIG03.TAB

ORANIG03.TAB is the 2003 version of the ORANI-G model (as currently used in the Practical GE Modelling Course[1] held each year at Monash University).

In ORANIG03.TAB, summary information is written to the logical file SUMMARY. ORANIG03.TAB contains statements such as:

Code in ORANIG03.TAB

```
Set INCMAC # Income Aggregates #
 (Land, Labour, Capital,  IndirectTax);
Coefficient (all,i,INCMAC)  INCGDP(i) # Income Aggregates #;
Formula
 INCGDP("Land")        = V1LND_I;
 INCGDP("Labour")      = V1LAB_IO;
 INCGDP("Capital")     = V1CAP_I;
 INCGDP("IndirectTax") = V0TAX_CSI;
Write INCGDP to file SUMMARY header "IMAC";
```

These statements collect together the aggregates that make up GDP from the income side V1LND_I, V1LAB_IO, V1CAP_I and V0TAX_CSI into one Coefficient INCGDP and write the values to the SUMMARY file. These are calculated from the pre-simulation data on the BASEDATA file.

As an example, the simulation in ORNG03SR.CMF is a 5 percent cut in the real wage using the short-run closure.[2] The following table shows INCGDP values on the SUMMARY file SUMMARY.HAR produced when you carry out this simulation. The values shown reflect the values in the original data OZDAT934.HAR.

Initial INCGDP values from original data file

IMAC	INCGDP
Land	2735.663
Labour	194981.218
Capital	180575.218
IndirectTax	49512.164
Total	427804.250

With Release 9, you can easily arrange for the calculation of summary facts based on the updated data to happen as part of the original simulation. All you have to do is to duplicate the write statements, as shown below.

For example, you could add the following statements:

Postsim Code in OG03PS.TAB

```
File (New) SUMMARY_UPD    # Summary info for updated data #;
Write (Postsim) INCGDP to file SUMMARY_UPD header "IMAC" ;
```

[1] See http://www.monash.edu.au/policy/pgemc.htm

[2] To carry out the simulation in ORNG03SR.CMF, first implement the ORANIG03 model by running TABLO, taking inputs from the Stored-input file OG03GS.STI. Then run GEMSIM taking inputs from the Command file ORNG03SR.CMF. [You will also need the data file OZDAT934.HAR.]

We have added these statements to ORANIG03.TAB to produce the new file **OG03PS.TAB**.[3]

Because the Write statement above includes the qualifier **(Postsim)**, the values of the GDP aggregates INCGDP written to file Summary_UPD will be the **updated or post-simulation** values.

We suggest that you carry out the wage-cut simulation using OG03PS.TAB. To do so,

- first implement OG03PS.TAB by running TABLO taking inputs from the Stored-input file **OG03PSGS.STI**. This produces output for GEMSIM.
- then carry out the simulation using Command file **WAGEPS.CMF**.[4] This has the same shock as in ORNG03SR.CMF but it is set up to carry out the simulation accurately (Gragg 2,4,6) based on OG03PS.TAB.
- Then look at the two Summary files produced.
- The first one is the file with logical name SUMMARY. The version of this file produced during the WAGEPS.CMF simulation is called **WAGEPS-SUM.HAR** since the Command file statement is
  ```
  file Summary = <cmf>-sum.har ;
  ```
  This file contains INCGDP values based on the pre-simulation data.
- The second one is the file with logical name Summary_UPD. The version of this file produced during the WAGEPS.CMF simulation is called **WAGEPS-SUM-UPD.HAR** since the Command file statement is
  ```
  file Summary_UPD = <cmf>-sum-upd.har ;
  ```
  This file contains INCGPD values based on the post-simulation data (since there is a "Postsim" qualifier in the Write statement added to OG03PS.TAB).

The INCGDP values in WAGEPS-SUM.HAR should be the same as the Initial INCGDP shown above.

The INCGDP values in WAGEPS-SUM-UPD.HAR should be as shown below.

Post-simulation values of INCGDP

IMAC	INCGDP
Land	2940.36
Labour	185162.02
Capital	180363.03
IndirectTax	49623.88
Total	418089.28

For example, you can see that the value of INCGDP("labour") falls by roughly 5 percent from 194981 in the original data to 185162 in the updated data.

How is the post-sim value of INCGDP(*"Labour"*) calculated?

> Initially V1LAB values are read in from file BASEDATA. Hence after the simulation, V1LAB(i,o) has the same values as the updated values written to the updated version of file BASEDATA. The post-simulation values of the coefficients V1LAB_O, V1LAB_IO and INCGDP(*"Labour"*) are then calculated from these updated V1LAB(i,o) values using the Formulas:
>
> ```
> Formula (all,i,IND) V1LAB_O(i) = sum{o,OCC, V1LAB(i,o)};
> Formula V1LAB_IO  = sum{i,IND, V1LAB_O(i)};
> Formula INCGDP("Labour") = V1LAB_IO;
> ```

Before Release 9, if you wanted to know the same summary facts about the updated data as you calculated about the pre-simulation data, you had to re-run the program GEMSIM (or ORANIG03) starting from the updated data.[5]

---

[3] We use "PS" in the name to indicate that there are Post-sim sections in the code. In fact we have added several sections of post-sim processing code in OG03PS.TAB. These are all at the end of OG03PS.TAB. The other post-sim sections are discussed later in this chapter.

[4] You will also need the data file OZDAT934.HAR.

If you calculate INCGDP using the updated data from the ORNG03SR.CMF simulation modified to use Gragg 2,4,6 steps, you should get the same values as are shown in the table above showing the Post-simulation values of INCGDP. However to calculate these values, you have to run the program GEMSIM (or ORANIG03) twice, once to run the wage-cut simulation and get the updated data, and then a second time to calculate the summary values from the updated data.

2.1.2 Adding Some Post-sim Processing to GTAP61.TAB

In GTAP, a summary of the salient features of the database is not written as part of the main GTAP model but is handled by various related TAB files such as GTAPVIEW.TAB and DECOMP.TAB.

However with Release 9 of GEMPACK, summaries of both the original and updated coefficients can be written in the main TAB file. For example, in the TAB file GTAP61.TAB, the coefficient GDP(r) contains values of the Gross Domestic Product in region r. GDP(r) is calculated by the formula below:

Formula for GDP in GTAP61.TAB

```
Coefficient (all,r,REG)
    GDP(r) # Gross Domestic Product in region r #;
Formula (all,s,REG)
    GDP(s) = sum(i,TRAD_COMM, VPA(i,s) )
           + sum(i,TRAD_COMM, VGA(i,s) )
           + sum(k,CGDS_COMM, VOA(k,s) )
           + sum(i,TRAD_COMM, sum(r,REG, VXWD(i,s,r)))
           + sum(m,MARG_COMM, VST(m,s))
           - sum(i,TRAD_COMM, sum(r,REG, VIWS(i,r,s)));
```

In order to write the pre-simulation and also the post-simulation values of GDP(r), you could introduce two summary files and write the coefficient values as shown below: We have added these statement at the end of GTAP61.TAB to produce **GT61PS.TAB**.

Postsim Code in GT61PS.TAB

```
File (new) SUMMARY  #Summary file for the original data #;
File (new) SUMMARY_UPD  #Summary file for the updated data #;
Write GDP to file SUMMARY header "GDP" ;
Write(PostSim) GDP to file SUMMARY_UPD header "GDP" ;
```

To see how this works, we have included in the Example files the Command file **GSEF1PS.CMF**. This is the same simulation as in the standard Command file GSEFTA1.CMF but modified to work with GT61PS.TAB. GSEF1PS.CMF includes the following two statements

```
file summary = <cmf>-sum.har ;
file summary_upd = <cmf>-sum-upd.har ;
```

To see the effect of these changes,

- first implement GT61PS.TAB by running TABLO taking inputs from the Stored-input file **GT61PSGS.STI**. This produces output for GEMSIM.
- then carry out the simulation by running GEMSIM taking inputs from the Command file **GSEF1PS.CMF**.[6]

Then look at the two Summary files produced.

---

[5] The Windows programs RunGTAP and RunGEM enable users to see these summaries for the updated data. They do so by running the second job on the updated data in the background before they report that the simulation is finished.

[6] You will also need the data files GDATA7X5.HAR, GSETA7X5.HAR and GPARA7X5.HAR, and the shock file GTMSA7X5.SHK.

- The first one is the file with logical name SUMMARY. The version of this file produced during the GSEF1PS.CMF simulation is called **GSEF1PS-SUM.HAR** (see the Command file statements above). This file contains GDP values based on the pre-simulation data.
- The second one is the file with logical name Summary_UPD. The version of this file produced during the GSEF1PS.CMF simulation is called **GSEF1PS-SUM-UPD.HAR** (see the Command file statements above). This file contains GPD values based in the post-simulation data (since there is a "Postsim" qualifier in the Write statement added to GT61PS.TAB).

You can see, for example, that the pre-simulation value of GDP in SAFRICA is about 135338 while the post-simulation value of GDP in SAFRICA is about 136560.

2.1.3 New File Can Contain Both the Original and Updated Data

You can also write the original pre-simulation data and the updated or post-simulation data to the same file. To do this, you can declare a new File (that is, one to which you will write the values of Coefficients or Variables) in either the normal or the Postsim part of the TAB file.

Such a new file can contain both pre-simulation and post-simulation values. Continuing the GTAP61 example from the previous section, to write both the pre-sim and post-sim values of GDP to the same file, you can add the following statements to the GTAP61.TAB file. We have added these statements to produce **`GT61P2.TAB`**.

Postsim Code in GT61P2.TAB

```
File (New) Pre_Post # Holds Pre-sim and Post-sim values # ;
Write GDP to file Pre_Post header "GDP"
   Longname "Original GDP values";
Write(PostSim) GDP to file Pre_Post header "GDPU"
   Longname "Updated GDP values" ;
```

To see how this works, we have included in the Example files the Command file **GSEF1P2.CMF**. This is the same simulation as in the standard Command file GSEFTA1.CMF but modified to work with GT61P2.TAB. GSEF1P2.CMF includes the following statement

```
file Pre_Post = <cmf>-pre-post.har ;
```

To see the effect of these changes,

- first implement GT61P2.TAB by running TABLO taking inputs from the Stored-input file **GT61P2GS.STI**. This produces output for GEMSIM.
- then carry out the simulation by running GEMSIM taking inputs from the Command file **GSEF1P2.CMF**.[7]

Then look at the Pre_Post file produced. This has name **GSEF1P2-PRE-POST.HAR** because of the Command file statement above. If you look at this file in ViewHAR, you can see, for example, that the pre-simulation value of GDP in SAFRICA is about 135338 [header "GDP"] while the post-simulation value of GDP in SAFRICA is about 136560 [header "GDPU"]. Of course these values are the same as you saw for the same simulation (with slightly different TAB and Command files) in section 2.1.2 above.

[7] You will also need the data files GDATA7X5.HAR, GSETA7X5.HAR and GPARA7X5.HAR, and the shock file GTMSA7X5.SHK.

2.2 Examples – Post-Simulation Processing of Variable Results

In most models, it is necessary to do some post-simulation processing of results for the Variables. We give some examples below and show how this is simpler with the Release 9 post-simulation processing TAB file statements. Again we encourage you to carry out the examples below on your computer as you read about them.

2.2.1 GTAP – Table of Important Regional Results

Consider the GTAP model. You may wish to build up a table showing the results for some important variables in each region. Prior to Release 9, the program SLTOHT could be used to do this – see the first example (for variables u, y and EV) in section 9.4 of the Release 8 version of GPD-4.

With the Release 9 software, this table can be set up by adding to the GTAP TAB file the following section. We have added these statements near the end of the file **GT61PS.TAB** which is included with the Release 9 Examples files (see also section 2.1.2 above). Note the Formulas with Variables **u**, **y** and **EV** on the right-hand side and Coefficients on the left-hand side. The Coefficient RegResults1 is only used during the post-simulation processing.

Postsim Code in GT61PS.TAB

```
PostSim (Begin) ;  ! Beginning of a section of post-sim code !
SET RegVar1 (u, y, EV) ;
Coefficient (All,r,REG)(All,v,RegVar1) RegResults1(r,v) ;
Formula (All,r,REG) RegResults1(r,"u") = u(r) ;
Formula (All,r,REG) RegResults1(r,"y") = y(r) ;
Formula (All,r,REG) RegResults1(r,"EV") = EV(r) ;
File (new, text, SSE) PostSim1 ; !sse or spreadsheet!
Write RegResults1 to File PostSim1 ;
! or write it to a Header Array file !
PostSim (End) ;  ! Marks end of a section of post-sim code !
```

This shows a section of code which starts with the statement
PostSim (begin) ;
and ends with the statement
PostSim (end) ;

This is what we mean by a **postsim section of code** in a TAB file. The statements in this section are not carried out until after the simulation has been solved. Then the formulas and write statements in this postsim section of code are carried out during what we refer to as **post-simulation processing**.

The file with logical name PostSim1 holds the output produced in the Post-sim stage. The File statement for PostSim1
File (new, text, SSE) PostSim1;
uses the file qualifier **SSE** described in section 9.1 and produces output with row and column labels, in Comma Separated Value (CSV) format. (Spreadsheet programs can read CSV files.)

We have included in the Command file GSEF1PS.CMF (see section 2.1.2 above) the statement

file postsim1 = <cmf>-psim1.csv ;

You ran the GSEF1PS.CMF simulation in section 2.1.2 above. If you look at the CSV file produced (it is called **GSEF1PS-PSIM1.CSV**) in your spreadsheet program, you can see the table shown below.

SSE Output from GSEF1PS.CMF

RegResults1(REG:RegVar1)	u	y	EV
SAFRICA	0.264	0.958	306.319
RESTSAF	–0.113	–0.089	–17.436
RESTSSH	–0.005	–0.005	–7.762
EUNION	–0.002	–0.005	–131.458
RESTWLD	–0.001	–0.001	–115.048

2.2.2 TERM – Table of Important Regional Results

TERM is a regional model of a single country (such as Australia or China) developed by Mark Horridge and colleagues at the Centre of Policy Studies, Monash University. See

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

for more details.

The TERM model organises the main macro results (by region) into a variable called MainMacro(m,q) where index m ranges over various interesting macro variables (real household expenditure, real investment etc) and index q ranges over the regions in the model. It is used mainly to build up a table of regional macro results.

Code in TERM.TAB

```
Set MAINMACROS # Set of reporting main macro variables #
(RealHou, RealInv, RealGov, ExpVol, ImpVolUsed, ImpsLanded, RealGDP,
   AggEmploy, averealwage, AggCapStock,CPI,ExportPI, ImpsLandedPI,
   Population);
Variable (all,m,MAINMACROS)(all,q,REG) MainMacro(m,q)
         # Convenient table of macros for reporting #;
Equation
 E_MainMacroA (all,q,REG) MainMacro("RealHou",q) = xfin("Hou",q);
 E_MainMacroB (all,q,REG) MainMacro("RealInv",q) = xfin("Inv",q);
 E_MainMacroE (all,q,REG) MainMacro("ImpVolUsed",q) = ximpused(q);
! and so on!
```

With Release 9 software, this could also be done in a post-simulation part via Coefficients and Formulas rather than using Variables and Equations. We have added the following post-sim section to TERM.TAB to produce **TERMPS.TAB**. In this code, the Coefficient called MainMacroF is given the same values as are given in the code above to the Variable MainMacro. The Coefficient MainMacroF is written to the spreadsheet file with logical name PostSim1. As a check the code below also writes the values of the Variable MainMacro to this file. Of course these values should be identical to those of MainMacroF.

Postsim Code in TERMPS.TAB

```
PostSim (Begin) ;  ! The beginning of a section of post-sim code !
Coefficient (all,m,MAINMACROS)(all,q,REG) MainMacroF(m,q)
            # Convenient table of macros for reporting #;
Formula
 (all,q,REG) MainMacroF("RealHou",q) = xfin("Hou",q);
 (all,q,REG) MainMacroF("RealInv",q) = xfin("Inv",q);
 (all,q,REG) MainMacroF("ImpVolUsed",q) = ximpused(q);
! and so on!
File (new, text, sse) PostSim1 ;
Write MainMacroF to file PostSim1 ; ! or to a Header Array file !
! Also write MainMacro – values should be same as MainMacroF. !
Write MainMacro to file PostSim1 ;
PostSim (End) ;  ! Marks end of a section of post-sim processing !
```

We suggest that you run a simulation with TERMPS.TAB to see how this works first hand. [The relevant files are all supplied with Release 9 of GEMPACK.] To do so,

- first run TABLO taking inputs from Stored-input file **TERMPSGS.STI**. This produces output for GEMSIM.
- then run GEMSIM taking inputs from Command file **TERM1PS.CMF**.[8] This carries out a simulation in which government spending in region Gippsland is increased by 10 percent.

[8] You will also need the data files AGGMOD.HAR and AGGSETS.HAR.

The simulation produces a spreadsheet file called **TERM1PS.CSV** (since the relevant statement in TERM1PS.CMF is "File PostSim1 = <cmf>.csv ;"). You can load this file into your favourite spreadsheet program (or into TABmate) and look at the results. The table of results for MainMacroF should be similar to that shown below. Notice that the table has element labelling. That is because the qualifier "SSE" (see section 9.1 below) is used when declaring the file PostSim1 in TERMPS.TAB.

MainMacroF(MAINMACROS:REG)	Gippsland	OtherVic	OtherAus
RealHou	1.3859	1.47E-03	-1.39E-02
RealInv	1.095918	-4.42E-03	-2.10E-02
RealGov	10	0	0
ExpVol	-2.30686	-5.87E-02	-6.41E-03
ImpVolUsed	0.75452	4.21E-03	-5.26E-03
ImpsLanded	0.976353	2.46E-03	2.82E-03
RealGDP	0.721188	8.06E-03	-1.83E-03
AggEmploy	1.39802	1.35E-02	-1.88E-03
averealwage	1.96E-08	-1.23E-09	2.53E-11
AggCapStock	0	0	0
GDPPI	0.85818	1.75E-02	-7.32E-03
CPI	0.807638	1.57E-02	-5.34E-03
ExportPI	0.465686	1.17E-02	1.28E-03
ImpsLandedPI	0	0	0
Population	0	0	0

2.2.3 ORANIG03 – Table of Winning and Losing Industries

In post-simulation sections, you can use variables containing the results of the simulation in formulas and carry out post-simulation processing of your results.

In the ORANI-G model, variable **x1tot(i)** reports the percentage change in the activity level in industry i. With Release 9 software, it is easy to add post-simulation code to the TABLO file to report the winning and losing sectors from any simulation. The code is shown below. We have included this code near the end of **OG03PS.TAB**.

The code starts with the statement
```
PostSim (begin) ;
```
and ends with
```
PostSim(end) ;
```
Some new coefficients are defined. Then Formulas are used to manipulate **simulation results for x1tot.**

Note the use of the functions MAXS and MINS (see section 4.4.2 of the Release 8 version of GPD-2) to work out the maximum and minimum x1tot values.

The sets Win_Ind and Lose_Ind are data-dependent sets where, in this case, the data are the x1tot simulation results. The Variable x1tot is used just like a Coefficient in the Post-simulation code. If you used x1tot in Data-dependent sets like Win_Ind and Lose_Ind or in write statements in the ordinary part of the TAB file, there would be a syntax error. However Variables can be used like Coefficients in a post-simulation part.

Finally in the code below, the x1tot results for the winning sectors (those in set Win_Ind) and for the losing sectors (those in set Lose_Ind) are written to different headers on the output Header Array file with logical name Postsim1.

Postsim Code in OG03PS.TAB

```
Postsim (begin) ;  ! Work out the winning and losing sectors !
Coefficient MAX_X1TOT ;
Coefficient MEAN_X1TOT ;
Coefficient MIN_X1TOT ;
Coefficient NIND # Number of industries # ;
Formula NIND = SUM(i,IND,1) ;

Formula  MAX_X1TOT = MAXS(i,IND,x1tot(i)) ;
Formula  MIN_X1TOT = MINS(i,IND,x1tot(i)) ;
Formula  MEAN_X1TOT = SUM(i,IND,x1tot(i))/NIND ;

Set Win_Ind # Winning industries, judged by x1tot results #
  = (all,i,IND: x1tot(i) >= (MAX_X1TOT+MEAN_X1TOT)/2 ) ;

Set Lose_Ind # losing industries, judged by x1tot results #
  = (all,i,IND: x1tot(i) <= (MIN_X1TOT+MEAN_X1TOT)/2 ) ;

Write x1tot to file Summary_UPD Header "XTOT" ;
Write (all,i,Win_Ind) x1tot(i) to file Summary_UPD
 Header "WINI"
  longname "Winning industries, judged by x1tot results" ;
Write (all,i,Lose_Ind) x1tot(i) to file Summary_UPD
 Header "LOSI"
  longname "Losing industries, judged by x1tot results" ;
Postsim (end) ;
```

To see how this code works, look at the file **WAGEPS-SUM-UPD.HAR** produced when you ran the wage-cut simulation with OG03PS.TAB and WAGEPS.CMF in section 2.1.1 above.

- You can see the x1tot results at header "XTOT".
- You can see the x1tot results for the winning industries at header "WINI". Note that 10 of the 35 industries are deemed to be "winners" using the test in the TAB code above.[9]
- You can see the x1tot results for the losing industries at header "LOSI". Note that, by coincidence, 10 of the 35 industries are deemed to be "losers" using the test in the TAB code above.[10]

[9] From the code in OG03PS.TAB, an industry is a "winner" if its x1tot value lies is higher than the value halfway between X1TOT_MEAN and X1TOT_MAX. If you look in the LOG file WAGEPS.LOG produced by the WAGEPS.CMF simulation, you will see that X1TOT_MIN = –0.0913, X1TOT_MAX = 4.87 and X1TOT_MEAN = 2.36. [These values are written to the LOG file via the xwrite statements in a Post-sim section (see section 2.6) of the Command file WAGEPS.CMF.] Thus an industry is in the winning set if its x1tot value is greater than (X1TOT_MEAN+X1TOT_MAX)/2 which is about 3.62.

[10] If you look at the winners and losers from the TARCUTPS.CMF simulation discussed in section 2.2.4 below, you will find that 21 industries are deemed to be winners and only one industry a loser. In chapter 3, we introduce another way of finding winners and losers, and they are ranked in order, as you will see when you get to that chapter.

2.2.4 ORANIG03 - Tariff cut Simulation Results

Another of the simulations covered in the Practical GE Modelling Course[11] held at Monash University each year is a reduction in tariffs on imported Clothing and Footwear, using the ORANI-G model of Australia.

In analysing this simulation, you may wish to know the tariff rate TARFRATE on different commodities and other related data such as V0TAR, V0CIF and V0IMP **both pre-sim and post-sim**.

It may also be helpful to consider some of the tariff-related variables calculated during the simulation. These variables are on the Solution file but it may be convenient to add some of these variables, for example, t0imp and delV0TAR, to the post-sim summary file.

We have included the following code segment near the bottom of **OG03PS.TAB** to show this sort of post-simulation information whenever you carry out a simulation with this version of ORANI-G.

Postsim Code in OG03PS.TAB

```
File (New) SUMMARY_UPD       # Summary info for updated data # ;

! Save the pre-sim values so can calculate percent changes post-sim !
Coefficient(parameter) (All,c,COM) V0TARORIG(c) ;
Formula(initial) (All,c,COM) V0TARORIG(c) = V0TAR(c) ;
Coefficient(parameter) V0TAR_CORIG ;
Formula(initial) V0TAR_CORIG = V0TAR_C ;

PostSim(Begin);
Write  ! Same headers as pre-sim values written to SUMMARY file !
  TARFRATE to file SUMMARY_UPD header "TRAT";
  V0TAR    to file SUMMARY_UPD header "0TAR";
  V0CIF    to file SUMMARY_UPD header "0CIF";
  V0IMP    to file SUMMARY_UPD header "0IMP";
  t0imp to file SUMMARY_UPD header "t0im" ;
  delV0TAR to file SUMMARY_UPD header "dTAR" ;
! percent change = change / original value * 100 !
Coefficient (All,c,COM) p_V0TAR(c) #percent change in V0TAR#;
Formula (All,c,COM)
p_V0TAR(c)=IF(V0TARORIG(c) NE 0.0, 100*delV0TAR(c)/V0TARORIG(c)) ;
Write  p_V0TAR to file SUMMARY_UPD header "pTAR" ;
Coefficient p_V0TAR_C #percent change in total V0TAR_C# ;
Formula p_V0TAR_C = 100*Sum(c,COM, delV0TAR(c))/V0TAR_CORIG ;
Write p_V0TAR_C to file SUMMARY_UPD header "V0TC" ;
PostSim(End) ;
```

The code above includes calculation of the percentage change values p_V0TAR(c) using the simulation variable delV0TAR and the pre-simulation values V0TARORIG of V0TAR. The change variable delV0TAR is used in the model because some of the V0TAR values are zero. However you can calculate p_V0TAR using the PostSim formula:

```
Formula (All,c,COM)
p_V0TAR(c)=IF(V0TARORIG(c) NE 0.0, 100*delV0TAR(c)/V0TARORIG(c)) ;
```

We suggest that you carry out the tariff-cut simulation to see the effects of the statements above. To do this,

- first run TABLO taking inputs from Stored-input file **OG03PSGS.STI**. This produces output for GEMSIM.

[11] See http://www.monash.edu.au/policy/pgemc.htm

- then run GEMSIM taking inputs from Command file **TARCUTPS.CMF**.[12] This carries out a simulation in which government spending in region Gippsland is increased by 10 percent.

You can look at the pre-sim tariff values in the SUMMARY file which is called **TARCUTPS-SUM.HAR** and at the post-sim tariff values in the Summary_UPD file which is called **TARCUTPS-SUM-UPD.HAR**. You will see, for example, that

> the pre-simulation value of TARFRATE("ClothingFtw") is 0.1956 (which means an *ad valorem* tariff rate of 19.56%) while the post-simulation value of TARFRATE("ClothingFtw") is 0.0760 (which means an *ad valorem* rate of 7.60%). [See header "TRAT" in the relevant files.]

2.3 Sophisticated Post-Simulation Processing

Various standard models carry out sophisticated post-simulation processing. They typically do this via TAB files which are run after the simulation is completed. The program SLTOHT is used to convert the simulation results to a Header Array SOL file. The post-simulation processing TAB files read simulation results from this SOL file.

Excellent post-simulation processing tools of this kind have been developed for the GTAP model. These are currently automated under the RunGTAP interface which handles the running of the different jobs (SLTOHT, DECOMP, GTAPVOL etc).

Using Release 9 software, all of this sort of post-simulation processing could be done more simply in the TAB file for the main model.

Below we illustrate this by showing in section 2.3.1 how the code in various typical parts of DECOMP.TAB could be replaced by code in GTAP.TAB.[13]

2.3.1 GTAP – Sophisticated Post-Simulation Processing in DECOMP.TAB

Associated with the GTAP model are several TAB files that process and organise simulation results. Examples are DECOMP.TAB and GTAPVOL.TAB. These are run after the simulation has been completed (and after SLTOHT has been run to convert the simulation results on the Solution file to a Header Array file), and carry out very sophisticated post-simulation processing.

Using Release 9 software, all of the work which is done in DECOMP.TAB could be done more simply in the TAB file GTAP.TAB for the main model.[14] And the results currently obtained from DECOMP.TAB would then be available every time the main model is solved (rather than having to first run SLTOHT and then run DECOMP as is done now).

Below we show how this could be done. We look at some typical parts of DECOMP.TAB and show how these could be replaced by post-simulation processing code which could be added to GTAP.TAB.[15]

---

[12] You will also need the data file OZDAT934.HAR.

[13] We supply with the GEMPACK examples files versions of DECOMP.TAB and GTPVOL.TAB (both dating from around January 2000) for you to look at if you wish while reading this section. This section is not a hands-on section, just a discussion about what might be. You should not attempt to run the DECOMP.TAB or GTPVOL.TAB (the name currently used for GTAPVOL.TAB under RunGTAP) supplied. If you want to work with these and with the GTAP model, you should obtain the current GTAP.TAB, DECOMP.TAB and GTPVOL.TAB from the GTAP web site.

[14] For example, the postsim code shown later in this section could be added to GTAP61.TAB supplied with the GEMPACK examples.

[15] We have not supplied with the Release 9 examples a version of GTAP.TAB which includes the post-simulation processing sections we suggest below. One day these might become part of the standard GTAP.TAB.

Code Organising Variable Results into Welfare Decomposition Headers

There are several parts of DECOMP.TAB whose purpose is simply to organise the results of variables into appropriate headers for the welfare decomposition.

Here is a typical example of such code.

Code in DECOMP.TAB

```
File GTAPSOL # file containing all solution variables #;
Coefficient (all,r,reg)   cntalleffr(r)
  # total contribution to regional EV of allocative effects #;
Read   cntalleffr from file GTAPSOL header "ALLR";
Coefficient (all,r,reg)(all,j,column)
    welfare(r,j) # aggregate report of welfare change #;
Formula (all,r,reg)  welfare(r,"alloc_A1") = cntalleffr(r);
! and so on to fill all of Coefficient welfare !
Write welfare to file WELVIEW header "A"
    longname "summarized welfare report";
```

The simulation results are put onto the file connected to logical filename GTAPSOL by running SLTOHT before DECOMP is run. In Release 9, there would be no need to run SLTOHT. The above code could be replaced by the following code (in a post-simulation part of GTAP.TAB).

Postsim Code which could be added to GTAP.TAB

```
PostSim (Begin) ;
Coefficient (all,r,reg)(all,j,column)
    welfare(r,j) # aggregate report of welfare change #;
Formula (all,r,reg)  welfare(r,"alloc_A1") = cntalleffr(r);
! and so on to fill all of Coefficient welfare !
Write welfare to file WELVIEW header "A"
    longname "summarized welfare report";
PostSim (End) ;
```

Note that the results in Variable **cntalleffr** can be accessed directly in this postsim section of code.

Code Accessing Pre- and Post-Simulation Values of Coefficients

DECOMP.TAB also needs to access pre-simulation values of Coefficients (for example BVOA and BVOM below) and post-simulation values of the same Coefficients (for example, UVOA and UVOM below). These are calculated using values of things like BVFM (base values) and UVFM (updated values) which are read from files GTAPBASE and GTAPUPDATE respectively.

Code in DECOMP.TAB

```
File  GTAPBASE # file containing all base data #;
File  GTAPUPDATE # file containing all updated base data #;
Coefficient (all,i,ENDW_COMM)(all,j,PROD_COMM)(all,r,REG) BVFM(i,j,r)
   # Base producer expenditure on i by j in r at market prices #;
Read      BVFM from file GTAPBASE header "VFM";
Coefficient (all,i,NSAV_COMM)(all,r,REG)  BVOM(i,r)
   # Base value of commodity i output in region r at mkt prices #;
Formula (all,i,ENDW_COMM)(all,r,REG)
    BVOM(i,r) = sum(j,PROD_COMM, BVFM(i,j,r));
! Then formulas for BVOM(i,r) for i in other sets (MARG_COMM etc) !
! Similarly for BVOA(i,r) – Base values of VOA !
Coefficient (all,i,ENDW_COMM)(all,j,PROD_COMM)(all,r,REG) UVFM(i,j,r)
  # Updated producer expenditure on i by j in r at market prices #;
Read     UVFM from file GTAPUPDATE header "VFM";
Coefficient (all,i,NSAV_COMM)(all,r,REG)  UVOM(i,r)
   # Updated value of commodity i output in region r at mkt prices #;
Formula (all,i,ENDW_COMM)(all,r,REG)
    UVOM(i,r) = sum(j,PROD_COMM, UVFM(i,j,r));
! Then formulas for UVOM(i,r) for i in other sets (MARG_COMM etc) !
! Similarly for UVOA(i,r) – Updated values of VOA !
! Then are formulas below calculating pre-sim tax rates BOUTAX
   and post-sim tax rates UOUTAX using BVOA,BVOM and UVOA,UVOM !
Zerodivide (nonzero_by_zero) default 0;
Coefficient (all,i,NSAV_COMM)(all,r,REG)  BOUTAX(i,r);
Formula (Initial) (all,i,NSAV_COMM)(all,r,REG)
    BOUTAX(i,r) = [BVOM(i,r) - BVOA(i,r)] / BVOM(i,r) * 100;
Coefficient (all,i,NSAV_COMM)(all,r,REG)  UOUTAX(i,r);
Formula (all,i,NSAV_COMM)(all,r,REG)
    UOUTAX(i,r) = [UVOM(i,r) - UVOA(i,r)] / UVOM(i,r) * 100;
Zerodivide (nonzero_by_zero) off;
Coefficient (all,i,NSAV_COMM)(all,r,REG)(all,c,COL) OUTPUT(i,r,c) ;
Formula (all,i,NSAV_COMM)(all,r,REG)  OUTPUT(i,r,"taxrateb") = BOUTAX(i,r);
Formula (all,i,NSAV_COMM)(all,r,REG)  OUTPUT(i,r,"taxrateu") = UOUTAX(i,r);
Write OUTPUT to file WELVIEW header "A211"
 longname "combination of output welcnt,vol change and tax rate";
```

Using Release 9 software, this could all be done in GTAP.TAB. The Formula for BOUTAX could be done as a Formula(Initial) in a non-postsim part. The right-hand side of this Formula can refer to VOM and VOA (since these are equal to the Base values in a Formula(Initial) in a non-postsim part). There is no need for Coefficients BVOA and BVOM to be declared explicitly. Of course the Formula(Initial) for BOUTAX must be placed after the values of Coefficients VOA and VOM have been calculated via the usual Formulas in GTAP.TAB. Nor would there be any need for the Coefficients UVOA and UVOM to be declared explicitly (and hence no need of File GTAPUPDATE). If VOA and VOM are accessed in a post-simulation part of the code (see the Formula for UOUTAX below), their values are automatically the post-simulation (or, updated) values. That is, the following code could be added to GTAP.TAB to achieve what is done in the section of DECOMP.TAB code shown above. Note that the first part (calculating BOUTAX) must not be in a post-sim part while the second part (calculating UOUTAX) must be in a post-sim part.

Code which could be added to GTAP.TAB

```
! These formulas setting base values do not go in a post-sim part !
 Zerodivide (nonzero_by_zero) default 0;
Coefficient (Parameter) (all,i,NSAV_COMM)(all,r,REG) BOUTAX(i,r)
  # Base value of output tax rates # ;
Formula (Initial) (all,i,NSAV_COMM)(all,r,REG)
! VOM,VOA here are base values since is Formula(Initial) !
    BOUTAX(i,r) = [VOM(i,r) - VOA(i,r)] / VOM(i,r) * 100;
Zerodivide (nonzero_by_zero) off;

! Next must be in a post-sim part !
PostSim (Begin) ;
Zerodivide (nonzero_by_zero) default 0;
Coefficient (all,i,NSAV_COMM)(all,r,REG) UOUTAX(i,r)
  # Updated value of output tax rates # ;
Formula (all,i,NSAV_COMM)(all,r,REG)
! VOM,VOA here are the updated values since in post-sim part !
    UOUTAX(i,r) = [VOM(i,r) - VOA(i,r)] / VOM(i,r) * 100;
Zerodivide (nonzero_by_zero) off;
Coefficient (all,i,NSAV_COMM)(all,r,reg)(all,c,col) OUTPUT(i,r,c) ;
Formula (all,i,NSAV_COMM)(all,r,reg)
   OUTPUT(i,r,"taxrateb") = BOUTAX(i,r);
Formula (all,i,NSAV_COMM)(all,r,reg)
   OUTPUT(i,r,"taxrateu") = UOUTAX(i,r);
Write OUTPUT to file WELVIEW header "A211"
 longname "combination of output welcnt,vol change and tax rate";
PostSim (End) ;
```

GTAPVOL.TAB

Another part of the post-simulation processing supplied with the GTAP model is GTAPVOL.TAB. This has code similar to the above examples from DECOMP.TAB. All of the processing done via GTAPVOL.TAB could be done more simply in post-simulation parts of GTAP.TAB.

2.3.2 Other Models

Sophisticated post-simulation processing could be set up easily for other models in the main TAB file for those models. Then you won't need a fancy Windows interface such as RunGTAP for your model. Every time you carry out a simulation, the post-solution processing will be done at the same time. We say a little more about this in section 2.4 below.

2.4 Advantages of Post-Sim Processing in TAB File – Overview

Most current post-simulation processing involving simulation results and/or updated data requires separate jobs and separate TAB files. By using Post-Sim processing, you can avoid running separate jobs after the main simulation. Instead you carry out the required Post-Sim processing as part of the main simulation.

- **Past:** Typically SLTOHT is run to convert simulation results in the Solution file to a Header Array file or to produce tables in a spreadsheet file.
  Now: The tables can be added in a post-simulation part of the TAB file for the main model. [See the examples in sections 2.1 and 2.2 above.]
- **Past:** TABLO-generated programs corresponding to separate TAB files (for example DECOMP.TAB and GTAPVOL.TAB) may be run. These separate TAB files duplicate the READ statements and many of the Formulas in the TAB file for the main model. This causes maintenance issues when the model changes.
  Now: This sophisticated post-simulation processing can be done in the TAB file for the main model, and the results are available immediately once the simulation is run. See section 2.3 above from more details.
- **Past:** Windows programs such as RunGTAP and RunGEM have been written to automate the running of SLTOHT and the running of TABLO-generated programs based on the updated data or reading the simulation results.
  Now: Whatever your model, you will be able to carry out simple or sophisticated post-simulation processing every time you solve your model. You won't need to commission Mark Horridge or Ken Pearson to write a customised Windows interface for your model. All you will need to do is to add post-simulation processing parts to the TAB file for your main model. Then every time you carry out a simulation, the post-solution processing will be done at the same time.

2.5 New TAB File Statements, Syntax and Semantic Rules

Post-simulation processing can be done in **post-simulation sections** of the TAB file. There are two ways of writing post-simulation sections:

1. You can begin a post-simulation section with the statement

   ```
   PostSim (Begin) ;
   ```

   and end with the statement

   ```
   PostSim (End) ;
   ```

   The PostSim statements above mark the beginning and end of a post-simulation section of the TAB file. There can be several such post-sim sections in a single TAB file. The statements that can be used in post-simulation sections are listed in section 2.5.1.

2. Alternatively you can add the qualifier **(PostSim)** to any **Write** or **Display** statement anywhere in the TAB file to indicate that this statement relates to the post-simulation part of the TAB file.[16] For example, to write or display the post-simulation values of GDP which is calculated by a Formula in the ordinary part[17] of the TAB file (see section 2.1.2), you can add the statements

   ```
   Write (PostSim) GDP to file SSOutput ;
   Display (PostSim) GDP ;
   ```

You should think of all PostSim sections of the TAB file being in order at the end of the file, and think of all PostSim statements being done in order in a single pass after the equations of the model have been solved.[18] Of course, a post-simulation section is only allowed in TAB files able to carry out simulations. You cannot do post-simulation processing in data-manipulation TAB files or in TAB files with Equations but no Update statements.

We often need to distinguish between

- the **post-simulation part of a TAB file**. This is the part consisting of all postsim sections of the TAB file. [This also includes all Write and Display statements with a (Postsim) qualifier.]
- the **ordinary or normal part of a TAB file**. This is all the non-postsim parts of the TAB file.

2.5.1 Statements Allowed in Post-Simulation Sections of a TAB File

The statements allowed in post-simulation sections of a TAB file (between a *PostSim (begin) ;* statement and a *PostSim(end) ;* statement) are as follows.

- **Set** and **Subset** statements. You can define new Sets and Subsets which are only needed in the post-simulation part.
- **Coefficient**. You can define Coefficients which are only needed in the post-simulation part. We call these **postsim Coefficients** to distinguish them from Coefficients declared in the ordinary part of the TAB file (which we call **ordinary or normal Coefficients**).
- **File**. You can define new (logical) Files which are only needed in the post-simulation part.
- **Read**. You can read in extra outside data. You can only read the values of postsim Coefficients. You cannot read into Variables or into normal Coefficients in postsim sections of the TAB file.

---

[16] Formally each Write or Display statement with a (Postsim) qualifier is a post-sim section consisting of a single statement.

[17] The **ordinary part** of the TAB file means the non-post-simulation part.

[18] In fact the postsim statements are done in two passes at the end – see section 2.8.

- **Formula**. You can write Formulas to calculate the values of postsim Coefficients (on the left-hand side of the Formula). In postsim Formulas, Variables and Coefficients are allowed on the right-hand side. When a normal Coefficient is used on the right-hand side, its values are the post-simulation values (if relevant). Variables cannot be on the left-hand side. Coefficients declared in the ordinary (that is, not-post-simulation) part of the TAB file cannot be on the left-hand side. See section 2.5.2 below for more details.
- **ZeroDivide**. Separate defaults apply in postsim sections from the normal sections of the TAB file. See section 2.5.5 below for details.
- **Mapping**. You can specify Set Mappings which can be used in the post-simulation part.
- **Write**. You can write any Coefficient or Variable to Header Array, text or spreadsheet files. Post-sim values of normal Coefficients are written when there is a Write statement in a postsim part of the TAB file. Writes to GAMS files are not allowed in the post-simulation part.
- **Display**. You can display any Coefficient or Variable. The values are added at the end of the Display file (see section 4.3 of GPD-3). Post-sim values of normal Coefficients are written to the Display file when there is a Display statement in a postsim part of the TAB file.
- **Assertion**. Coefficients and Variables are allowed. The values given to a normal Coefficient are its post-simulation values. See section 2.5.4 below for more details.

The statements in the post-simulation part of the TAB file are processed in the order in which they appear in the TAB file.

You cannot include Variable, Equation, Update, Transfer, Omit, Substitute, Backsolve or Complementarity statements in a post-simulation part of the TAB file.

- We use the adjective **Postsim** to refer to things declared in a Postsim part of the TAB file. For example, we refer to "Postsim sets" and "Postsim Coefficients".
- We use the adjective **normal or ordinary** to refer to things declared in a normal (that is, not a Postsim) part of the TAB file. For example, we refer to "normal sets", "ordinary Coefficients" and "normal Coefficients".

2.5.2 Syntax and Semantics for Formulas in Postsim Part of TAB File

You can declare Coefficients that are only used in the post-simulation part. [As indicated earlier, we call these **postsim Coefficients** to distinguish them from Coefficients declared in the ordinary part of the TAB file.] You give these Coefficients values which can depend on the simulation results (values of Variables) and on the post-simulation values of other Coefficients. The syntax rules for Formulas in the post-simulation part are the same as for Formulas in the ordinary part except that Variables can be used on the right-hand side of Formulas (much as they can be used in Update statements).

The idea is that you can work with, **but not change**, the simulation results (Variables) and the post-simulation values of Coefficients. Variables and Coefficients from the ordinary part of the TAB file are allowed on the right-hand side of Formulas. If a Coefficient declared in the ordinary part of the TAB file is used on the right-hand side of a Formula in the postsim part, its values are the **post-simulation values**.

You can also work with (but not change) **pre-simulation values** of Coefficients if you set up these values by Formula(Initial)s in the non-post-simulation part of the TAB file.

Example

Postsim Code that could be added to GTAP.TAB

```
Coefficient (all,r,REG) SAVE(r)
# expenditure on NET savings in r valued at agent's prices #;
Update (all,r,REG)  SAVE(r) = psave(r) * qsave(r);
Read   SAVE from file GTAPDATA header "SAVE";
! Above statements are in standard GTAP.TAB !

Coefficient(parameter) (all,r,REG) SAVEORIG(r)
   # pre-sim value of SAVE # ;
Formula (Initial) (all,r,REG) SAVEORIG(r) = SAVE(r) ;

Display (postsim) SAVE ;
Display (postsim) SAVEORIG ;
```

If Coefficient SAVE is used in the post-simulation part, its values are the post-simulation values. The
`Display(postsim) SAVE ;`
statement will display the postsim values of SAVE.

If Coefficient SAVEORIG is used in the post-simulation part, its values are the pre-simulation values. The
`Display(postsim) SAVEORIG ;`
statement will display the pre-simulation values of SAVE.

You cannot put a Variable on the left-hand side of a Formula. [This is because you are not allowed to change the simulation results.]

You cannot put an ordinary Coefficient (that is, a Coefficient declared in the ordinary part of the TAB file) on the left-hand side of a Formula in a postsim section of the TAB file. [This is because you are not allowed to change the values of "ordinary" Coefficients – their values in postsim sections are their post-simulation values.]

The values of other postsim Coefficients can change during the post-simulation part. For example, you can reuse a postsim Coefficient for holding results of intermediate calculations.

2.5.3 Semantics for Reads in Postsim Part of TAB File

These follow from the similar rules for Formulas.

- You cannot read into a Variable. [This is because you are not allowed to change the simulation results.]
- You cannot read values into a Coefficient that is declared in the ordinary part of the TAB file. [This is because you are not allowed to change the values of ordinary Coefficients – their values in postsim parts of the TAB file are always equal to their post-simulation values.]
- You cannot do both normal and postsim reads from the same file. [You can split the data file into two parts, putting the data read in the postsim part of the TAB file onto a separate file if this is a problem.][19]

2.5.4 Semantics for Assertions in Postsim Part of TAB File

These follow from the similar rules for Formulas.

[19] We have prohibited both normal and postsim reads from the same file mainly to simplify our code. If both sorts of reads were allowed from the same file, there would be problems in organising the code to put the data read in the postsim part on the updated version of the file. And there would be problems for the user in anticipating the order of reads if the file is a text file.

In the post-simulation part of the TAB file, if a Coefficient from the ordinary part of the TAB file is used in the logical expression containing the condition, its values are the post-simulation values. [If you have set up Coefficients to hold pre-simulation values (see section 2.5.2 above), these can also be used in Assertions.]

Example

Postsim Code that could be added to GTAP.TAB

```
Postsim (begin) ;
Assertion # expenditure on NET savings < 100000 post-sim #
 (all,r,REG) SAVE(r) < 100000 ;

Assertion # Pre-sim NET savings >= Post-sim Net Savings #
 (all,r,REG) SAVEORIG(r) >= SAVE(r) ;
PostSim (end) ;
```

2.5.5 Semantics for ZeroDivides in Postsim Part of TAB File

As indicated earlier, you can think that all statements in the postsim part of the TAB file are done in order in a single pass at the end of the run.[20] At the beginning of this pass, the Zerodivide status is the same as at the beginning of the TAB file (see section 4.13 of GPD-2), namely

- division of zero divided by zero is allowed and the result is zero.
- division of a nonzero by zero is not allowed.

During a postsim pass, only Zerodivide statements in the postsim part affect the results. Any Zerodivide statements in the normal part of the TAB file have no effect during a postsim pass.

Example

```
ZeroDivide (NonZero_By_Zero) default 6 ;
ZeroDivide (Zero_By_Zero) Default 3 ;
Postsim (Begin) ;
Coefficient coef1 ;
Formula coef1 = 0/0 ;
Write coef1 to terminal ;
Coefficient coef2 ;
Formula coef2 = 3/0 ;
Write coef1 to terminal ;
Postsim (End) ;
```

If this is the first postsim section in the TAB file, the value of coef1 will be zero (not 3) and the formula for coef2 will result in an error at run time, since division of a nonzero by zero is not allowed at that stage in the postsim part. (If this is not the first postsim part in the TAB file, the ZeroDivide defaults could have been set in a previous postsim section.)

If you want to use Zerodivide statements in the Postsim part, it is safer and clearer to set them within the postsim section where they are used (and then turn them off again within the same postsim section).

2.5.6 Default Statements and Certain Qualifiers Not Relevant in Postsim Part

There is no need to distinguish between Coefficient (Parameter) and Coefficient (Non_Parameter) in the postsim part, since no updates are made of Coefficients during post-simulation processing. Nor is there any need to distinguish between Formula (Initial) and Formula (Always). Conceptually, the

[20] Actually the postsim statements are done in two postsim passes – see section 2.8. What we say here applies to both of those passes.

values of all postsim Coefficients are only calculated once and all postsim Formulas are only carried out once.[21]

For this reasons default statements (see section 3.20 of GPD-2) are not allowed in the postsim part of the TAB file. For example, the statements

```
Coefficient (Default=Parameter) ;  ! Not allowed in postsim part !
Formula (Default=Initial);         ! Not allowed in postsim part !
```

are not allowed in a postsim section.

Similarly, the qualifiers Parameter or Non_Parameter are not relevant when a Coefficient is declared in a postsim part. These qualifiers are ignored in the postsim part.

Nor are the qualifiers Initial or Always relevant in Formulas in the postsim part. These qualifiers are ignored in the postsim part.

2.5.7 Post-Sim Sets, Subsets, Coefficients not Allowed in the Ordinary Part

You cannot use Sets, Subsets, Set Mappings or Coefficients declared in the post-simulation part of the TAB file in statements in the ordinary part of the file (even if the post-simulation sets or coefficients appear earlier in the TAB file).

Example

```
Set COM (c1-c10) ;
Postsim (Begin) ;
Set COM1 (c1-c3) ;
Coefficient (All,c,COM1) COEF1(c) ;
Coefficient (All,c,COM) COEF2(c) ;
Postsim (End) ;
Coefficient (All,c,COM1) COEF3(c) ; ! NO – set COM1 not allowed here !
Coefficient C4 ;
Formula C4 = SUM(c,COM,COEF2(c)) ; ! NO – Coeff COEF2 not allowed here !
```

2.5.8 PostSim Qualifier for Write or Display

You can add the qualifier **(PostSim)** to a Write or to a Display statement anywhere in the TAB file. This can be useful if you have a single post-sim write or display statement amongst other ordinary statements. See the ORANIG03 and GTAP examples in section 2.1.

2.5.9 Implications for Condensation

You cannot omit or substitute out any Variable that appears in the post-simulation part since its values will not be available. However you can backsolve for such Variables.

[21] If they affect a set, some formulas may be carried out on both of postsim passes two and three (see section 2.8). But the coefficients in question have the same values on these two passes.

2.6 Postsim Extra Statements in a Command File

Some TABLO-like statements are allowed in a Command file – the so-called "extra" statements on a Command file, as documented in section 6.6 of GPD-3.

You can include Post-simulation versions of these extra statements on your Command file. This is only relevant for extra **`Set, Subset, File, Write`** and **`Display`** statements.[22]

You can have a group of such postsim statements enclosed between

```
XPostsim (Begin) ;
```

```
XPostsim (End) ;
```

statements. [As usual, an "X" is required the front of all TABLO-like statements on the Command file.] For example,

```
XPostsim (Begin) ;
XSet COM4 (c1-c4) ;
XSubset COM4 is subset of COM ;
Xfile (New) Postsim2 ;
XWrite (all,c,COM4) Coeff1(c) to file Postsim2 header "CF1" ;
Xdisplay x1 ; ! Displaying values of a variable
XPostsim (End) ;
```

You can also use a "Postsim" qualifier for single postsim writes or displays as in[23]

```
Xwrite (Postsim) Coeff1 to terminal ;
Xdisplay (Postsim) x1 ;  ! Displaying values of a variable
```

2.7 Suppressing Post-Simulation Processing

If there are Postsim statements in your TAB file (or extra postsim statements in your Command file), you can suppress the corresponding post-simulation processing when the program runs by including the statement

```
postsim = no ;   ! default is "yes"
```

in your Command file.

[22] The other sorts of Postsim statements (for example, Read, Coefficient, Formula) are not allowed as normal or postsim extra statements on Command files.

[23] As with other extra statements (see section 6.6.2 of GPD-3), make sure that there is a space after the keyword before the "(" which indicates the beginning of the qualifier.

2.8 When and How Is Post-Simulation Part Done?

The TABLO-generated program or GEMSIM proceeds as normal to solve the simulation. It runs through all subintervals (if you have specified more than one), extrapolates (if requested), writes the simulation results to the Solution file and writes the updated data, all as usual.

Then, if there is a post-simulation part, the software pretends to start an extra subinterval. There are three postsim passes.

- On the **first postsim pass**, just the Reads and Formulas in the **ordinary part** of the TAB file are carried out. The Read statements cause values to be read from the updated file so that the values of Coefficients read are post-simulation values. All Formulas in the normal part of the TAB file are carried out. However Formula(initials)s are not done. Instead the post-sim values of the Coefficient calculated initially by Formula(initial)s are read from a temporary updated file.[24]

  At the end of this pass, the values held in all normal Coefficients are the updated values. No write or display statements are carried out during this pass.

  Each Assertion(Always) statement in the ordinary part of the TAB file is carried out on this pass. The updated values of all Coefficients are used.

- On the **second postsim pass**, the Sets, Subsets and Set Mappings in the **post-simulation part** are organised. This is similar to the preliminary pass at the start of the run (see section 13.2 of GPD-3) when the Sets, Subsets and Set Mappings in the ordinary part of the TAB file are organised.

- On the **third postsim pass**, the Reads, Formulas, Assertions, Writes, Displays and ZeroDivides in the **post-simulation part** are carried out. For this purpose, you can think of all postsim sections of the TAB file being in order at the end of the file, and that all postsim statements are done in order in this single pass at the end.

  At the end of this pass, post-simulation values are held in all Coefficients, both normal and Postsim.

At the end of the third postsim pass, the program ends. You can view the simulation results via ViewSOL, the updated data (via ViewHAR) or analyse results (via AnalyseGE – see section 2.9) as usual. You also have the post-simulation results available immediately. You can load these into Excel, Word or ViewHAR.

We refer to the three passes described above as **postsim passes**.

We refer you to some examples in the section below.

[24] For this reason, postsim values of a Coefficients initialized by a Formula(initial) may be different from the values calculated if you start from the updated data – see the example in section 8.5.2 of GPD-3.

2.8.1 Example – SJPS.TAB and SJPSLB.CMF

Included amongst the examples distributed with Release 9 of GEMPACK are SJPS.TAB and SJPSLB.CMF.

SJPS.TAB consists of the standard SJ.TAB (see chapters 2 and 3 of GPD-1) with the following section of postsim processing code added at the end. This produces a table of sectoral results.

Postsim Processing Code in SJPS.TAB

```
PostSim (Begin) ;
File (New, Text, SSE) Results1 # holds results for reporting # ;
Set SECTRES (PC, XCOM, XH) ;
Coefficient (All,i,SECT)(All,v,SECTRES) SECTRESULTS(i,v) ;
Formula (All,i,SECT) SECTRESULTS(i,"PC") = p_PC(i) ;
Formula (All,i,SECT) SECTRESULTS(i,"XCOM") = p_XCOM(i) ;
Formula (All,i,SECT) SECTRESULTS(i,"XH") = p_XH(i) ;
Write SECTRESULTS to file Results1 ;
PostSim (End) ;
```

SJPSLB.CMF is the standard Command file SJLB.CMF (see chapter 2 of GPD-1) modified to work with SJPS.TAB. This Command file SJPSLB.CMF carries out the standard 10% increase in labor simulation. It produces the following table of sectoral results.

SECTRESULTS(SECT:SECTRES)	PC	XCOM	XH
s1	0	5.88527	5.88527
s2	-0.94858	6.899293	6.899292

We suggest that you carry out this simulation via the following steps.

- First run TABLO taking inputs from the Stored-input file **SJPSGS.STI**. That will produce output for GEMSIM.
- Then run GEMSIM taking inputs from the Command file **SJPSLB.CMF**.[25]

Check that the output Results1 file called **SJPSLB.CSV** contains the table of results shown above.

Then look at the output LOG file **SJPSLB.LOG** in TABmate. You should be able to see the usual steps for carrying out the simulation. Then, towards the end of the LOG file, you should see where the three postsim passes described in section 2.8 above are carried out.

Search for the words "**Starting the Post-sim Job**". This marks the beginning of the processing of the postsim part of the TAB file. You will see the following lines.

```
 ---> Starting the Post-sim Job

 --->  Beginning pass number 1 of 3-pass calculation.

 [On this pass, do all reads and formulas, starting from the updated data.
  This finds the post-simulation values of all Coefficients.]

(Opened existing file 'sjpslb.upd'.)
```

These mark the start of the first postsim pass. Note that the reading of values (for example, those of DVCOMIN) is done from the updated data file SJPSLB.UPD.

Notice that all the normal Formulas are done during this pass. These calculate the postsim values of the relevant Coefficients.

At the end of this pass, notice that the SLC file is written. When postsim processing is done, the SLC file contains pre-simulation values of the Coefficients (as it did in Release 8) and also (this is new for

[25] You will also need the data file SJ.DAT.

Release 9) the post-simulation values of the same Coefficients. For example, if you open SJPSLB.SLC in ViewHAR, you will see the pre-simulation values of Coefficient XC at header 0007 and the post-simulation (or updated) values of Coefficient XC at header U007.

Next in the LOG file SJPSLB.LOG you will see the lines

```
  --->  Beginning pass number 2 of 3-pass calculation.

  [On this pass, do all sets, subsets and mappings
   defined in the PostSim part of the TAB file.]

 Set 'SECTRES'
   [Set has size 3.]
```

These constitute the whole of the second postsim pass. This is the pass in which postsim Sets, Subsets and Set Mappings are worked out. Here only the set SECTRES needs to be worked out. [Note that this set is not worked out on the preliminary pass since it is a postsim set.]

Next in the LOG file you will see the lines

```
--->  Beginning pass number 3 of 3-pass calculation.

  [On this pass, carry out all actions (reads, formulas, writes etc)
   in the PostSim part of the TAB file.]

  Formula for 'SECTRESULTS'
  Formula for 'SECTRESULTS'
  Formula for 'SECTRESULTS'

 (Opened, for writing, file 'sjpslb.csv'.)
 Writing 'SECTRESULTS'
 (Written real array, size 2x3.)
```

This is where the values of postsim Coefficient SECTRESULTS are calculated and written.

Finally, you will see in the LOG file the lines

```
  (Completing the Solution Coefficients (SLC) file.)

  [Finished the Post-sim Job.]
```

The program waits until the end of all postsim passes to write the details of sets and subsets to the SLC file. This is because it needs to write the postsim sets as well as the ordinary sets.

That completes all the postsim processing and the program then concludes as usual.

2.8.2 Other Examples

In earlier sections of this chapter, you carried out several simulations which include post-simulation processing. You can view the LOG files from these simulations in order to see what processing was done on each of the 3 postsim passes.

2.9 Postsim Parts of TAB File and AnalyseGE

When you load a Solution file and associated SLC file into AnalyseGE[26], if the TAB file for the model contains any Postsim statements[27], the SLC file contains the pre-simulation and the post-simulation values of all normal Coefficients (as distinct from Postsim coefficients) – see section 9.5.1. Then you can easily use AnalyseGE to see the pre-simulation or the post-simulation values of Coefficients or expressions, as we describe below.

2.9.1 Coefficients or Expressions Selected from the TABmate Form

- If you ask to evaluate a postsim Coefficient (that is, a Coefficient declared in the postsim part of the TAB file) in a postsim statement, you will see its values. [These are the values of the Coefficient at the end of the third postsim pass.]
- If you ask to evaluate a normal Coefficient (one declared in a normal part of the TAB file), the values that you see depend on the choice you have made under the **AnalyseGE menu** item on the TABmate form.

  If menu item **Use Pre-sim Coeff Values** is selected, pre-simulation values will be used for any normal Coefficients in the expression, so the result is the pre-simulation value of the Coefficient.

  If menu item **Use Post-sim Coeff Values** is selected, post-simulation values will be used for any normal Coefficients in the expression so the result is the post-simulation value of the Coefficient.

  If menu item **Pre/Post-sim from TABLO Statement** is selected (this is the one selected every time you open a new solution in AnalyseGE), the values depend on whether or not the TABLO statement you have clicked on is a postsim one. If it is a normal (as distinct from a postsim) statement, the pre-simulation values of the Coefficient (which must be a normal Coefficient) are shown. If it is a postsim statement, the postsim (that is, updated) values of the Coefficient are shown.

- The same applies when you evaluate an expression.

  If menu item **Use Pre-sim Coeff Values** is selected, pre-simulation values will be used for any normal Coefficients in the expression, so the result is the pre-simulation value of the expression.

  If menu item **Use Post-sim Coeff Values** is selected, post-simulation values will be used for any normal Coefficients in the expression so the result is the post-simulation value of the expression.

  If menu item **Pre/Post-sim from TABLO Statement** is selected, the values depend on whether or not the TABLO statement you selected the expression from is a postsim one. If the expression comes from a normal (as distinct from a postsim) statement, the Coefficients (which must be normal Coefficients) are given their pre-simulation values when the expression is evaluated. If the expression comes from a postsim statement, any normal Coefficients in the expression are given their post-simulation (that is, updated) values when the expression is evaluated.

You can change the options setting under the AnalyseGE menu on the TABmate form several times during one run if you wish. For example, by doing that you can see the pre-simulation values of certain expressions and then later see the post-simulation values of the same expressions.

---

[26] This applies for Version 2.70 (March 2004) or later of AnalyseGE.

[27] Or if there are any Postsim sections in the Command file.

2.9.2 Expressions Evaluated from the AnalyseGE Form

On the AnalyseGE form there will be two Evaluate buttons **Evaluate Presim** and **Evaluate Postsim** when there are any Postsim statements in the TAB file, or when you load an SLC file which contains post-simulation values of some coefficients. You can use these buttons to evaluate either the pre-simulation or the post-simulation values of expressions you type into the Expression memo on the AnalyseGE form.

2.9.3 UDC Files Now Redundant?

It is now easy to ensure that both the pre-simulation and the post-simulation values of all normal coefficients are on the SLC file.

- These are all put onto the SLC file whenever there are any postsim statements in the TAB file or on the Command file – see section 9.5.1.[28]
- If you don't have any postsim statements in the TAB file, just including a single Postsim Write statement such as

  ```
  Xwrite (Postsim) <coefficient-name> to terminal ;
  ```

  in your Command file, where <coefficient-name> is the name of any coefficient in your TAB file, will ensure that the SLC file contains post-sim values as well as pre-sim values of all normal Coefficients.

When the SLC file contains post-simulation values as well as pre-simulation values, you can look at either in AnalyseGE (see sections 2.9.1 and 2.9.2 above). Hence the SLC file is at least as useful as the corresponding UDC file (which only contains post-simulation values). So you probably won't need to produce a UDC file again. [See section 8.5 of GPD-3 for information about UDC files.[29]]

2.10 Postsim Sets and Subsets not on Equations File

When GEMSIM or a TABLO-generated program writes an Equations file, no information about sets or subsets in the postsim part of the TAB file is written to the Equations file. Only information about the sets and subsets in the normal part of the TAB file is written. This is because the sets and subsets in the postsim part might depend on the simulation results. But information on the Equations file is unrelated to any simulation.

However information about sets and subsets in the postsim part of the TAB file is written to the Solution file.

---

[28] Unless you have the statement "postsim = no ;" in your Command file – see section 2.7.

[29] If you ask for a UDC file, it is written during Postsim pass 1, since the values of coefficients on the UDC file are just the postsim values. The extra step (for example, extra pass 3 after a 2-step Euler in a 2, 4, 6-step Euler) as described in section 8.5.1 of GPD-3 is not done for Release 9 or later.

CHAPTER 3

3. Ranking Sets via Data (Finding Winners And Losers)

You may wish to rank sets according to simulation results (for example, to identify the 5 industries which gain most and/or the five which lose most) or according to data or values of Coefficients (either pre-simulation or post-simulation values).

This is now easy to do since you can define new sets whose elements are the same as another set but which are ordered according to the values of the relevant simulation results or Coefficient.

First we give an example in section 3.1 which shows how you can ask TABLO to identify winners and losers. Then we give (in section 3.2) the general rules for these Ranked Set statements.

3.1 Identifying Winners and/or Losers

Suppose that **x0ind** is a variable in your model which ranges over the set IND of industries and which reports the percentage changes in industry output. Suppose that the simulation results are:

Industry	beef	wheat	mining	steel	manuf	services
x0ind result	0.2	1.3	-2.5	1.8	0.6	-4.1

You can reorder the elements of the set IND according to the **x0ind** results, using the statements below.

```
Set INDRUP = IND Ranked UP BY x0ind ;
```

```
Set INDRDOWN = IND Ranked DOWN BY x0ind ;
```

Since these statements rely on the results for Variable x0ind, they must be in a Postsim part of the TAB file (see chapter 2).

With the x0ind results as above, you can see that the sets are:

INDRUP = (services, mining, beef, manuf, wheat, steel)

INDRDOWN = (steel, wheat, manuf, beef, mining, services).

The first element of INDRUP is the element of IND for which x0ind has the smallest (here, the most negative) value. The last element of INDRUP is the element of IND for which x0ind has the largest value.

The first element of INDRDOWN is the element of IND for which x0ind has the largest value. The last element of INDRDOWN is the element of IND for which x0ind has the smallest (here, the most negative) value.

Of course, the elements of INDRDOWN are in the opposite order from those of INDRUP.

You can identify and put into a separate Coefficient (perhaps to go in a table in your report) the 3 biggest winners and/or the 2 biggest losers. To do that for the top 3 winners, include the statements

Code for Winners

```
PostSim (begin) ;
Set INDRDOWN = IND Ranked DOWN BY x0ind ;
Subset IND3WIN # top 3 winners according to x0ind results # =
   (all,i,INDRDOWN: $POS(i) <= 3) ;
Coefficient (all,i,INDRDOWN) x0indWIN(i) ;
Formula (all,i,INDRDOWN) x0indWIN(i) = x0ind(i) ;
PostSim (end) ;
```

and include the statements below for the biggest two losers.

Code for Losers

```
PostSim (begin) ;
Set INDRUP = IND Ranked UP BY x0ind ;
Coefficient (integer) NIND # Number of industries # ;
Formula NIND = SUM(i,IND, 1) ;
Subset IND2LOSE # worst 2 losers according to x0ind results # =
   (all,i,INDRDOWN: $POS(i) >= NIND-1) ;
Coefficient (all,i,INDRDOWN) x0indLOSE(i) ;
Formula (all,i,INDRDOWN) x0indLOSE(i) = x0ind(i) ;
PostSim (end);
```

Note the use of $POS (see section 4.4.4 of GPD-2) above, also the formula for calculating the size NIND of the set IND (see section 8.1 of GPD-2).

3.1.1 Example OG01PS.TAB

OG01PS.TAB is supplied with the Examples for Release 9. It is ORANIG01.TAB with the code shown on the next page added at the end to identify the top 5 winning industries.

The ORANI-G variable reporting percentage change in industry output is **x1tot**. You could rank the set IND using x1tot results (this would be similar to using x0ind as in the examples above). However, a large percentage change x1tot may occur in an industry which is very small (for example, as measured by its pre-simulation costs). A better way of ranking is by some approximation to the CHANGE (not percentage change) in industry activity. A good measure of that can be obtained by multiplying the pre-simulation level of activity (in the example below, we have used the pre-simulation value of costs V1TOT in each industry) by the percentage change x1tot in industry activity. We call the product CH_X1TOT in the TAB file.

In the part of the TAB file shown on the next page, you will see

- the Formula (Initial) for **V1TOTPRESIM** values. These are the pre-simulation values of costs V1TOT in each industry. Notice that this must be calculated outside the Postsim section of the TAB file. [If it were calculated inside the Postsim part, the values in V1TOT would be the post-simulation values, not the pre-simulation values – see section 2.5.]
- the Formula for **CH_X1TOT**. Notice that the variable x1tot can be treated as if it were a Coefficient in the Postsim part of the TAB file.
- the Set statement ranking IND using CH_X1TOT values. Here the ranking is DOWN which means that the industry with the largest CH_X1TOT value will be the first element of the ranked set **RankedIND**.
- the Set statement picking out the first 5 industries in the set RankedIND. The resulting set is called **Win5IND**. Note the use of $POS (see section 4.4.4 of GPD-2) in this Formula.
- the set **INDReportSet** used to assemble pertinent information about the winning 5 industries. [You could add further information about these industries here if you wished.]
- the Write statement to write out information about the 5 winning industries.

Statements in OG01PS.TAB to Identify Winners and Losers

```
! Post-simulation section to identify the "winning" industries !
! Use x1tot simulation results, weighted by pre-simulation
    V1TOT values, to indicate the winners. !
! Store pre-simulation values of V1TOT. This must be done in a
   normal part of this TAB file, not in the postsim part.
  And a Formula(Initial) must be used. [If a Formula(Always) is used,
   the values keep getting updated.] !

Coefficient (Parameter) (all,i,IND)  V1TOTPRESIM(i)
     # Pre-simulation total cost of industry i (including taxes) #;
Formula (Initial) (all,i,IND)  V1TOTPRESIM(i) = V1TOT(i) ;

Postsim (begin) ;

Coefficient (all,i,IND) CH_X1TOT(i)
  # measure of the change in real (not nominal) industry activity # ;
! Multiply the pre-simulation activity level as reflected by V1TOT
   by the percentage change x1tot in real industry activity.
 Here x1tot(i) is the simulation result but can be treated as a
 Coefficient since this is a Post-simulation part of the TAB file. !
Formula (all,i,IND) CH_X1TOT(i) = V1TOTPRESIM(i) * x1tot(i) ;

! Rank the set of industries by these CH_X1TOT numbers. !
Set RankedIND # Industries ranked down by change in real activity #
        = IND ranked down by CH_X1TOT ;
Set Win5IND # Winning 5 industries # =
       (all,i,RankedIND : $POS(i) <= 5) ;

File (new) PostSimRes # for post-simulation reporting # ;
Write (set) Win5IND to file PostSimRes header "WIN5" ;

Set INDReportSet (pchange, Value, ActChange) ;
Coefficient (all,i,IND)(all,r,INDReportSet) INDReport(i,r) ;
Formula (all,i,IND) INDReport(i,"pchange") = x1tot(i) ;
Formula (all,i,IND) INDReport(i,"Value") = V1TOTPRESIM(i) ;
Formula (all,i,IND) INDReport(i,"ActChange") = CH_X1TOT(i) ;


!write out the results for just the subset of industries Win5IND !
Write (all,i,Win5IND)(all,r,INDReportSet) INDReport(i,r)
   to file PostSimRes header "INDR" ;
Postsim (end) ;
```

WAGE Simulation OG01PS-WAGE.CMF

Supplied with OG01PS.TAB is the Command file OG01PS-WAGE.CMF which carries out a simulation in which real wages are reduced by 5 percent. We suggest that you carry out this simulation via the following steps.

- Run TABLO, taking inputs from the Stored-input file **OG01PSGS.STI**. This produces output for GEMSIM.
- Run GEMSIM taking inputs from the Command file **OG01PS-WAGE.CMF**.[30]

The results for the winning 5 industries will be in output file **OG01PS-WAGE-RES.HAR** since the relevant statement in the Command file is "File PostSimRes = <cmf>-res.har ;".

If you look in ViewHAR at the Coefficient INDReport(Win5IND,INDReportSet) written at header "INDR" on the PostSimRes output file OG01PS-WAGE-RES.HAR, you will see the following.

INDReport	pchange	Value	ActChange
1 Transport	8.398	33186.5	278682.8
2 Finance	3.514	45984.9	161595.2
3 FoodFibre	12.369	12721.1	157345.5
4 Trade	2.757	56489.9	155717.5
5 TrnspEquip	11.786	10734.6	126518.3

Notice that the largest x1tot result shown is for FoodFibre. But this only ranks third in the ActChange (CH_X1TOT) values because its base costs are smaller than those for the Transport and Finance industries which have smaller percentage changes in x1tot.

3.2 Ranked Set Statements – Syntax and Semantics

These are Set statements of the form[31]

```
SET <new-set> [#<labelling-info>#] =
          <old-set> Ranked UP|DOWN BY <coeff/var-name> ;
```

Here

- <new-set> is the name of the new set defined. It will be a non-intertemporal set.
- [#<labelling-info>#] is optional labelling information (as for other sets).
- <old-set> is the name of the old set whose elements are to be ranked or re-ordered in <new-set>. Note that <new-set> and <old-set> have the same elements though probably in a different order.
- <old-set> must be a non-intertemporal set whose elements will be known (at run time).
- you choose either UP or DOWN. If you choose UP, the elements of <new-set> are ordered so that the first element is the one for which <coeff/var> has the smallest value, the second element is the one for which <coeff/var> has the next smallest value and so on. If you choose DOWN, the first element of <new-set> is the element of <old-set> for which <coeff/var> has the largest value, the second element if <new-set> is the one for which <coeff-var> has the second largest value, and so on.
- <coeff/var> must be the name of a Coefficient or Variable which has exactly one argument which ranges over <old-set>.
- if this is in a Postsim part of the TAB file (or Command file – see section 2.6), <old-set> can be any set (postsim or otherwise) and <coeff/var> can be any Coefficient (postsim or otherwise) or any Variable.

[30] You will also need the data file OGOZD867.HAR.

[31] We are grateful to Mark Horridge for suggesting this statement.

- if this is in an ordinary (not a Postsim) part of the TAB file (or Command file – see section 2.6), <old-set> must be an ordinary set (not a Postsim one) and <coeff/var> must be an ordinary Coefficient. [<coeff/var> cannot be a Postsim Coefficient, nor can it be the name of a Variable.]

There are several examples in section 3.1 above. These examples make it clear how DOWN or UP influences the order of the elements in <new-set>.

Examples.

```
Set INDRUP = IND Ranked UP BY x0ind ;

Set INDRDOWN = IND Ranked DOWN BY x0ind ;

Set RankedIND # Industries ranked down by change in real activity #
      = IND ranked down by CH_X1TOT ;

Set COMDOWN # Commodities ranked down by domestic sales #
      = COM ranked down by SALES ;
```

3.3 Converting Non-Intertemporal Sets to Intertemporal Sets and Vice Versa

In conjunction with ranking sets, it may be useful to be able to convert a non-intertemporal set to an intertemporal one, or vice versa.

Example

Suppose that you have ranked a large set of households and then want to calculate the Lorenz curve for income. In order to do this, you need to rank the households according to income. Then, for each household in the ranked set, add its income to the sum of the incomes of all households with a lower income. The natural formula for the latter is something like:

```
Formula (all,h,RankedHouse)
       CUMINC(h) = CUMINC(h-1) + Income(h) ;
```

This formula is only allowed in GEMPACK if the set RankedHouse is an intertemporal one.

Note that the alternative formula:

```
Formula (all,h,RankedHouse)  CUMINC(h) =
   SUM[k,RankedHous: Income(K) <= Income(h) : Income(k) ] ;
```

(which does not require the set RankedHouse to be an intertemporal set) is not suitable since it would take immense amounts of CPU time for a large set of households – say for 10,000 or more which is not a silly number to be working with.

But a ranked set must be a non-intertemporal one – see section 3.2 above. So, for the desired formula – the one which involves CUMINC(h-1) – to be allowed, the non-intertemporal set RankedHouse must be converted to an intertemporal one.

Section 3.3.1 describes the TABLO statements which allow non-intertemporal sets to be converted to intertemporal sets and vice versa.

Section 3.3.2 elaborates on the example about calculating the Lorenz curve for household income. You will see how the TABLO statements in section 3.3.1 are used.

3.3.1 TABLO Statements

There are two statements. The first allows a non-intertemporal set to be converted to an intertemporal one. The second goes in the opposite direction.[32]

```
Set (Intertemporal) <new-set> = <old-non-intertemporal-set> ;
Set (Non_Intertemporal) <new-set> = <old-intertemporal-set> ;
```

- In each case, the qualifier "(Intertemporal)" or "(Non_Intertemporal)" is required.
- In the first case, <old-non-intertemporal-set> must be a non-intertemporal set whose elements are known at run time. The intertemporal set <new-set> will have the same elements.
- In the second case, the elements of the new non-intertemporal set <new-set> will be determined from those in the <old-intertemporal-set>. If <old-intertemporal-set> has intertemporal elements (see section 7.2.1 of GPD-2), the elements of <new-set> have the "[" and "]" in the names omitted. Otherwise the elements of <new-set> are the same as those of <old-intertemporal-set>.
- Unlike other set equality statements (see section 3.1.2 of GPD-2 and section 8.7.1 below), there are no Subset statements automatically generated by these statements.

Examples

```
Set COM (c1-c3) ;
Set (Intertemporal)
   COMI # intertemporal version of COM # = COM ;
Set (Intertemporal) YEARS (Y[2001] – Y[2020] ) ;
Set (Non_Intertemporal)
   YEARSNI  # non-intertemporal version of YEARS # = YEARS ;
Set (Intertemporal) House (h1-h20) ;
Set (Non_Intertemporal)
   HouseNI # non-intertemporal version of House # = House ;
```

Here

- COMI has elements (c1, c2, c3),
- YEARSNI has elements Y2001, Y2002 and so on until Y2020 (since YEARS has elements Y[2001], Y[2002] up to Y[2020]).
- HouseNI has elements h1, h2 up to h20.

The major difference between an intertemporal set and a non-intertemporal one is that you can use index offsets [for example, c-1, c+20] in formulas ranging over intertemporal sets but not over non-intertemporal sets.

3.3.2 Household Income Example in Detail

Here we elaborate on the Lorenz curve example in section 3.3 above. The relevant parts of the TAB file could be as shown on the next page.

You can use the files LORENZ.TAB, LORENZ.HAR and LORENZ.CMF to run this example. [These files are included amongst the examples distributed with GEMPACK.]

---

[32] As usual, labelling information is allowed after <new-set> in each case. Note that the statement

```
Set (Intertemporal) <int-set1> = <int-set2> ;   ! not allowed
```

is still not allowed if <int-set2> is an intertemporal set.

TABLO Code LORENZ.TAB for Calculating the Lorenz Curve for Household Income

```
File Input ;
Set House # Households #
  Read Elements from File Input Header "HOUS" ;
Coefficient (all,h,House) HousInc(h) # Household incomes # ;
Read HousInc from file Input Header "HINC" ;

Set RankedHouse # Households ranked by income #
                  = House Ranked Up by HousInc ;

Coefficient (all,h,RankedHouse)
     HousIncRank(h) # income in ranked order # ;
Formula (all,h,RankedHouse) HousIncRank(h) = HousInc(h) ;
Set RankedHouse2
  # ranked households except for one with lowest income # =
      (all,h,RankedHouse: $POS(h) > 1) ;

! Converting RankedHouse and RankedHouse2 to intertemporal sets !

Set (intertemporal)  RHouseINT
   # Intertemporal version of RankedHouse # = RankedHouse ;

Set (intertemporal)  RHouse2INT
   # Intertemporal version of RankedHouse2 # = RankedHouse2 ;

Subset RHouse2INT is subset of RHouseINT ;

Mapping RHINTtoRH from RHouseINT to RankedHouse ;
Formula (all,h,RHouseINT) RHINTtoRH(h) = $POS(h) ;

Coefficient (all,h,RHouseINT) HousIncInt(h)
   # income in ranked order over intertemporal set # ;
Formula (all,h,RHouseINT)
   HousIncInt(h) = HousIncRank(RHINTtoRH(h)) ;

! Calculate Lorenz curve !
Coefficient (all,h,RHouseINT) CUMINC(h) # Cumulative income # ;
! Next really only applies to household with lowest income.
  It will be overwritten just below for all other households. !
Formula (all,h,RHouseINT) CUMINC(h) = HousIncInt(h) ;
Formula (all,h,RHouse2INT)
  CUMINC(h) = CUMINC(h-1) + HousIncInt(h) ;

File (new) Output ;
Write HousInc to file Output Header "HOR" ;
Write HousIncInt to file Output Header "HRNK" ;
Write CUMINC to file Output Header "CUMI" ;
```

To see how this works, suppose that House has just 4 elements, namely h1-h4 and suppose that the HousInc data is:

Household	h1	h2	h3	h4
HousInc	35.0	26.0	32.0	30.0

Set RankedHouse = (h2, h4, h3, h1). [ranked up by HousInc]

Set RankedHouse2 = (h4, h3, h1). [RankedHouse without the first set element]

Set RHouseINT = (h2, h4, h3, h1) [same as RankedHouse but is an intertemporal set]

Set RHouse2INT = (h4, h3, h1) [same as RankedHouse2 but is an intertemporal set]

Mapping RHINTtoRH from RHouseINT to RankedHouse maps
h2 -> h2, h4 -> h4, h3 -> h3 and h1 -> h1.

```
Formula (all,h,RHouseINT) HousIncInt(h)=HousIncRank(RHINTtoRH(h));
```

HousIncInt("h2")=HousInc("h2")=26, HousIncInt("h4")=HousInc("h4")=30,
HousIncInt("h3")=HousInc("h3")=32 and HousIncInt("H1")=HousInc("h1")=35.

RHouseINT	h2	h4	h3	h1
HousIncInt	26.0	30.0	32.0	35.0

```
Formula (all,h,RHouseINT) CUMINC(h) = HousIncInt(h) ;
```

CUMINC("h2")=26, also CUMINC("h4")=30, CUMINC("h3")=32, CUMINC("h1")=35 but these last three will be overwritten by the formula below.

```
Formula (all,h,RHouse2INT)
  CUMINC(h) = CUMINC(h-1) + HousIncInt(h) ;
```

CUMINC("h4") = CUMINC("h2") + HousIncInt("h4") = 26+30 = 56,
CUMINC("h3") = CUMINC("h4") + HousIncInt("h3") =56+32 = 88,
CUMINC("h1") = CUMINC("h3") + HousIncInt("h1") =88+35 =123.

RHouseINT	h2	h4	h3	h1
CUMINC	26.0	56.0	88.0	123.0

CHAPTER 4

4. When Memory is Short (GEMSIM or TG-Programs)

This chapter applies to very large models when you do not have enough physical memory[33] on your computer to solve the model. [The model may still solve under the modern Windows operating systems because they can use virtual memory. However, then it may take a very long time (many hours) to solve the model.]

This chapter applies only to GEMSIM and TABLO-generated programs. It does not apply to SAGEM or any of the other GEMPACK programs.

Release 9.0 requires considerably less memory compared to Release 8.0. Considerable work has been done to reorganise the code to reduce the total amount of memory without affecting the speed. More details are given in section 4.2.11.

Memory sharing, as described in section 4.2, reduces the memory required even further, as various examples in section 4.2 show.

4.1 Memory Reports

GEMSIM and TABLO-generated programs now produce a fairly accurate estimate of the total memory needed. If this amount is more than (or, even close to) the total amount of physical memory on your computer, you should consider using the memory sharing option described in section 4.2 below.

These programs allocate all the memory they think they will need early in the run (after the preliminary pass). In addition, as more information becomes available about the number of nonzeros (in the Equations file or after the LU decomposition), they allocate memory for these nonzeros. This extra memory is allocated (via the code values called MMNZ, MMNZ1 and MMNZ2 – see sections 13.3 and 13.4 of GPD-3 and section 5 in this document).

Each time MMNZ, MMNZ1 or MMNZ2 is increased, a report is given showing the total memory required so far. A final report showing the total memory required by the whole run is given near the end of the run. [34]

These reports look as follows:

Total memory used so far for all arrays is approximately 23.4 megabytes.
[This includes memory relating to MMNZ=7100, MMNZ1=7100 and MMNZ2=6000.]
[Add about 5-10 megabytes for the memory used by the code.]
[If memory sharing was not being used, this would be approx 29.3 Mb.]

[The last line is only present if memory sharing is used.]

[33] The amount of physical memory is often referred to as the amount of RAM. You can check the amount of physical memory on your Windows PC by right clicking on ***My Computer*** and selecting **Properties**. The amount of RAM is shown on the "General" tab.

[34] The first of these reports will be just after the value for TGMEM1 and TGMEM2 are reported. [This report corresponds to MMNZ=MMNZ1=1.] The next report comes just before the submatrices are done, when MMNZ and MMNZ1 are increased.

4.2 Memory Sharing

By default, all memory that is allocated stays allocated during the whole run.

GEMSIM and TABLO-generated programs can now modify this default behaviour so that the total memory needed at any stage is less than it would otherwise be. This is done by freeing up memory that is not needed for the current part of the calculation.

Example

> When doing the LU decomposition, the program does not need to know the values of all the Coefficients. The memory for these can be freed during the LU decomposition. Before freeing this memory, the current values of the Coefficients must be written to a work file. After the LU decomposition is finished, the memory allocated to the nonzeros can be freed, then memory allocated for the Coefficients. Then the current values of the Coefficients (which are needed for the backsolve and the update) can be read from the work file. In this way, the nonzeros and the Coefficients are **sharing memory**.

We expect that users will only want to use this memory-sharing option if the model is too large to fit into the available physical memory without it.

To activate this memory sharing, put the statement in your Command file:

```
share_memory = yes ;
```

[The default is " share_memory = no ;" which means not to share memory.][35]

4.2.1 When Memory Sharing May Be Useful

Sharing memory is only useful in the following two cases:

1. when you are carrying out a simulation.
2. when you are saving an Equations file.

It is of no use with a data-manipulation TABLO Input file.

Memory sharing may save you running in virtual memory, which is very slow – see section 4.2.2.

Of course, if you have enough total memory to solve your model without using memory sharing, there is no reason to use memory sharing.

4.2.2 Running in Virtual Memory is Usually Very Slow

Running in virtual memory can be very slow. The GEMPACK memory sharing feature will be most useful if it saves you having to use virtual memory.

The first serious use of memory sharing was during 2003 when Peter Dixon and Maureen Rimmer had to use a laptop (while overseas) with 1Gb of RAM to solve their large USAGE model (see section 4.2.10). This model solved in less than one hour on their desktop PC which had 2Gb of RAM. But it did not solve over-night on their laptop because, as we found, it was then running in virtual memory. Fortunately this model required less than 1Gb of RAM when the GEMPACK memory sharing was used, so that, with memory sharing, the model solved in an hour or so on the laptop.[36]

---

[35] During beta testing for Release 9, there was no "=" in this statement (which made it different from the form of most other statements in Command files). As a courtesy to beta testers, the software also accepts the statements "share_memory yes|NO ;".

[36] Release 9.0 GEMSIM and TABLO-generated programs use less memory than the Release 8.0 versions (see section 4.2.11). This was also helpful in getting the memory required below 1 Gb.

4.2.3 Restrictions

If you are carrying out a simulation and you ask the program to share memory, you cannot also save an Equations file on the same run. [Instead, just create the Equations file on one run and carry out the simulation on a second run. You can use memory sharing on both of these runs.][37]

You cannot use memory sharing if you start from existing Equations and BCV files (see sections 12.4.2 and 9.2.1 of GPD-3).

4.2.4 What Arrays Share Memory

The arrays which can share memory are those described below.

- Arrays holding the nonzeros (for example, during the LU decomposition).
- Arrays holding the current values of the Coefficients.
- Arrays holding the Updated values of the Coefficients.
- Arrays holding the solution values of all the Variables (for the steps carried out so far).
- Many permanent book-keeping arrays used to store things such as
  - ❖ the different multi-step solutions prior to the current step or subinterval,
  - ❖ the position in the columns of the Equations Matrix (see section 2.13.1 of GPD-1) corresponding to the first component of each condensed variable.
- Various temporary working arrays needed for some part of the calculation.

When you ask for memory sharing, the programs juggle these different arrays in and out of memory to minimise the total amount of memory required at any one time. You do not need to know about the details.

4.2.5 Main Memory Saving Comes During LU Decomposition

For many large models, the memory needed for the LU decomposition is often more than the memory needed for all the other arrays mentioned above. This is because the number of nonzeros in the LU decomposition can increase very much faster than the sizes of the other arrays as the model becomes larger (for example, as the numbers of commodities or regions increases). We give some figures below for some standard models to illustrate this.

Thus, the main memory saving usually comes during the LU decomposition. At that time, the program endeavours to free all arrays not directly needed for the LU decomposition. This often reduces the total amount of memory by a significant fraction (say 20% or more).

[37] If you are carrying out a simulation, the simulation will often require more memory if you also ask to save an Equations file (especially if your model has a very large number of exogenous variables only a few of which are shocked in most runs). So, if memory is short, save an Equations file and do simulations on separate runs (even if you are not using memory sharing).

4.2.6 Memory Saving If Using MA48 and MMNZ2 is Smaller than MMNZ

This section is only relevant if you are using the MA48 routines (rather than the MA28 routines) for the LU decomposition.

For most models, the minimum value of MMNZ2 (see the section 5 in this document) required for the LU decomposition is often smaller than the minimum values of MMNZ and MMNZ1 (these are equal when you use MA48) required. There is a report near the end of the LOG file stating these minimum values.

If memory is tight, it is a good idea to set MMNZ2 smaller than MMNZ and MMNZ1 via suitable

```
Start with MMNZ = <value> ;
Start with MMNZ2 = <value> ;
```

statements in your Command file (see section 5.1).

> Example. The GC10E1.CMF simulation with GTAP61 (see section 6.4.3 of GPD-3) requires MMNZ and MMNZ1 to be at least about 202,000 and MMNZ2 to be at least about 126,000. For this simulation it would be appropriate to include the statements
>
> ```
> start with MMNZ = 210000 ;
> start with MMNZ2 = 130000 ;
> ```
>
> [Provided you have sufficient memory, it is usually a good idea to set these values slightly more than the minimum.]

4.2.7 CPU Time Penalty

When you are using memory sharing, the programs are spending time writing values to work files, so you would expect that they will run more slowly than when you do not use memory sharing.

With the large models for which memory sharing is relevant, the CPU time is usually dominated by the time taken for the LU decomposition.[38] This means that the total CPU time for the memory sharing version is not prohibitively more than when memory sharing is not used.[39] The extra CPU time taken for memory sharing is probably much less that that which would be taken if you have to run in virtual memory (see section 4.2.2) if you don't use memory sharing.

We report CPU times for some of the example models (see sections 4.2.9 and 4.2.10). In our experience, the CPU penalty for memory sharing is not usually more than 15% of the total CPU time.

Of course, if you have enough total memory to solve your model without using memory sharing, there is no reason to use memory sharing.

4.2.8 Large Work Files Are Needed

When you use memory sharing, the programs need to create large work files (to hold the values in arrays which are temporarily taken out of memory).

In particular, the Equations Work file (it has suffix .E2K) may become very large. During the submatrix stage, the program is generating the nonzeros to go in the Left-Hand Side Matrix. Normally these are held in memory. However, when memory sharing is being used, they nearly all go onto the Equations Work file.

---

[38] The GTAP aggregations reported in section 4.2.9 are good examples of this. For the 38x39 aggregation, the total CPU time reported for the 1-step numeraire simulation was about 386 seconds and the LU decomposition took about 375 of these seconds. For the 47x52 aggregation the corresponding figures were about 1543 and 1515 seconds while for the 57x66 aggregation these figures were about 9606 and 9550 seconds.

[39] Indeed, the time for the LU decomposition is very sensitive to all sorts of small factors, including the order in which the nonzeros go into memory in the first place. In some cases, the CPU time for the memory-sharing version was less than that when memory sharing was used. Also the CPU time taken for very large models is rather variable from run to run.

If there are 2 million nonzeros in the Left-Hand Side Matrix, this Equations Work file will be about 24 Mb in size (multiply the number of nonzeros by 12).

4.2.9 GTAP Examples

Below we consider three highly disaggregated versions of the GTAP model, namely 38x39 (38 regions and 39 tradeable commodities), 47x52 (47 regions and 52 tradeable commodities) and 57x66 (the full 57 regions and 66 tradeable commodities for the version 5 GTAP data base). Note that these disaggregations are much larger than most modellers would want to solve (or, probably, should want to solve). We use them here to illustrate the effects of memory sharing. We used version 6.1 (August 2001) of GTAP.TAB, and the GTAP61.STI usually associated with that version of GTAP.TAB, in the results reported below.

In the table below, CPU is measured on a Pentium 4 processor running at 2.4 GHz, on a PC with 2Gb of RAM running Windows XP. The TABLO-generated program was compiled with LF95 version 5.70c. The compiler options are those suggested for use by default with the LF95 fortran compiler – see GPD-6. The figures below relate to a 1-step Euler numeraire simulation (using MA48) with these data sets. The CPU figures reported for memory sharing were obtained using a debug option which ensures that the nonzeros are in memory in the same order in the sharing and non-sharing cases.[40]

GTAP	Rel 9.0 Memory (no share)	Rel 9.0 Memory (share)	Memory Release 8.0	Min MMNZ (MA48)[41]	Min MMNZ2 (MA48)	CPU (1-step)[42]
38x39	310 Mb	230 Mb	450 Mb	21.5 million	12.5 million	7 minutes[43]
47x52	1183 Mb	908 Mb	1625 Mb	99.2 million	50.5 million	26-30 minutes[44]
57x66	1594 Mb	1196 Mb	2250 Mb[45]	120 million	71.5 million	160 minutes[46]

[40] The CPU time for LU decomposition seems to be very sensitive to the slightest change in initial circumstances. We used debug option 125 (put "debug options = 125 ;" in the Command file) when testing sharing times to ensure that the nonzeros on the LHS go into memory in exactly the same order in the sharing case as the non-sharing case. Had we not done this, the CPU variation because of different orders in memory could have swamped any CPU difference because of time spent reading and writing work files in the memory sharing case. [In the 57x62 case, when we did not use debug option 125, we found that the CPU time in the memory sharing case went up from about 2.7 hours to about 10 hours and the minimum values for MMNZ/MMNZ2 went from about 120/72 million respectively to about 188/90 million. We think that this was just by chance and think that, in some other case, it might have gone down. However we are investigating further.]

[41] MA48A was doing compressions (see section 5.2.1) when these values were obtained. Minimum values may be considerably higher if compressions are not done.

[42] These CPU times vary from run to run – see the footnotes attached to CPU column for the different aggregations.

[43] One time about 6 minutes 35 seconds when not sharing memory and about 7 minutes 22 seconds when sharing memory. A second time, elapsed time was 6 minutes 28 seconds when not sharing and 7 minutes 2 seconds when sharing.

[44] One time elapsed time was 25 minutes 44 seconds when not sharing and 29 minutes 30 seconds when sharing. A second time, elapsed time was 27 minutes 23 seconds when not sharing and 27 minutes 13 seconds when sharing.

[45] We have not solved this since programs are limited to 2 Gb of memory at present on Windows PCs. However we have obtained the TGMEM1 value from a run and know the minimum MMNZ value from the Release 9 run.

[46] One time, elapsed time was about 160 minutes when not sharing memory and about 158 minutes when sharing memory. A second time, elapsed time was 150 minutes when not sharing and 160 minutes when sharing.

4.2.10 USAGE Examples

These examples relate to the USAGE-ITC (USA General Equilibrium model, US International Trade Commission version, developed at Monash by Peter Dixon and Maureen Rimmer). This is a version of the MONASH model for the economy of the United States of America.

The results reported below are for about 500 sectors. The results relate to a historical simulation (no subtotals) and a corresponding decomposition simulation (12 subtotals).[47]

In all cases below, all of MMNZ,MMNZ1,MMNZ2 were set to equal 25 million. [In fact, the minimum values necessary are about 23.5 million for the historical and 14.4 million for the decomposition.[48] The minimum MMNZ2 (Release 9.0) is about 22.6 million for the historical and 14.4 million (same as MMNZ) for the decomposition. Setting the MMNZs closer to these minimums would reduce the memory in all cases, especially for the decomposition.]

USAGE-ITC	Rel 9.0 Memory (no share)	Rel 9.0 Memory (share)	Memory Release 8.0	CPU (1-step)
Historical	690 Mb	490 Mb	950 Mb	10 minutes
Decomposition	1090 Mb	590 Mb	1500 Mb	10 minutes

As you can see, Release 9.0 requires considerably less memory compared to Release 8.0, and memory sharing reduces the memory required even further. Reducing the memory required for the decomposition simulation to below 1Gb was vital for solving this simulation in a reasonable time on a laptop – see section 4.2.2.

4.2.11 Less Total Memory is Used Than in Release 8.0

Even without memory sharing, the Release 9.0 versions of GEMSIM and TABLO-generated programs need less total memory than the Release 8.0 versions. Some of the code has been reorganised to reduce the total amount of memory without affecting the speed. The saving compared to Release 8.0 is often 5% or more.

Example 1. Consider the GC10E1.CMF simulation with GTAP61 (see section 6.4.3 of GPD-3).

- This needs about 6.8Mb of memory under Release 8.0.
- With Release 9.0, this requires only about 4.8Mb (even if memory sharing is not used) or only about 3.7Mb if memory sharing is used.

Example 2. Consider the numeraire simulations with GTAP documented in section 4.2.9. You can see that the Release 8 memory is considerably higher than that needed for Release 9 for all three aggregations reported there.

Example 3. The USAGE simulations reported in section 4.2.10.

[47] See the MONASH model book Dixon and Rimmer (2002) for details about these simulations. The decomposition simulation is a rerun under a different closure of the historical simulation. These simulations are usually run using RunMONASH to facilitate the closure change.

[48] MA48A was doing compressions (see section 5.2.1) when these values were obtained. Minimum values may be considerably higher if compressions are not done.

4.2.12 Technical Notes About Memory Sharing

When this memory sharing is in force,

- the values of all Coefficients are written to a work file after the Equations part of each step and the memory allocated to them is then released.
- substantial memory for the nonzeros is only allocated during the LU decomposition part of each step of a multi-step calculation. During the Equations part of each step, a very small value (probably 50,000) is used for MMNZ, MMNZ1 and MMNZ2 [unless there is a "start with MMNZ\MMNZ1 = <value> ;" statement in the Command file, when that <value> is used]. This means that the nonzeros in the submatrices are written to a Work file during the Equations part of each step.
- if you are using the MA48 routines (rather than the MA28 routines), the LU decomposition is done in two steps. First MA48A is called to analyse the LHS Matrix. Then MA48B is called to factorise this matrix. While MA48A is running, there is no point in setting MMNZ or MMNZ1 larger than MMNZ2. For this reason, the software sets all three equal to the minimum of the "start with MMNZ|MMNZ2" values set in the Command file just before MA48A is called. Then, just before MA48B is called, the software will reduce the size of MMNZ2 (to the number of nonzeros in the LHS matrix) and will increase MMNZ and MMNZ1 to the value in the "start with MMNZ" statement in your Command file. This means that the MMNZ memory needed for LU decomposition is the maximum of
  12*MMNZ2 (this is the MMNZ memory needed for MA48A), and
  8*MMNZ+4*LHSNZ (this is the MMNZ memory needed for MA48B – here LHSNZ is the number of nonzeros in the LHS matrix).
  Since the minimum MMNZ2 is often much smaller than the minimum MMNZ (see the examples in the section 4.2.6), this can result in significant memory saving.
- after the LU decomposition, memory for the nonzeros is released, memory is reallocated for the Coefficients and the values of the Coefficients are read from the work file which was written at the end of the Equations part of the step. This is done after the LU decomposition and before the backsolves (if any) and updates for the step.

Because MMNZ and MMNZ1 have small values before the LU decomposition, the software may need to iterate a couple of times on the first step to get sufficiently high values for MMNZ and MMNZ1 in order to complete the LU decomposition. Thus the LU decomposition may take a little longer on the first step when this memory sharing is used. However, the MMNZ and MMNZ1 values used for the LU decomposition during the first step are remembered, so this inefficiency is not repeated during subsequent steps (or subsequent subintervals). Getting sufficiently large values for MMNZ etc on the first step usually does not take much extra time because reallocation is now done within the MA48 routines – see section 5.2.

4.2.13 Technical Fine Print about TABLO-generated Programs

This section is a technical one. It is here mainly to remind the code developers. You will probably prefer to skip it.

This fine print only applies to TABLO-generated programs. It does not apply to GEMSIM.

To make it possible for this memory sharing to have maximum effect for TABLO-generated programs, these programs now declare some Coefficients differently.

The difference is for those Coefficients all of whose related sets have fixed size (that is, whose size is defined in the TABLO Input file).

> For example, all sets in Stylized Johansen (SJ.TAB) have fixed size but in GTAP61.TAB the sizes of most sets are only determined at run time.

Consider a Coefficient all of whose related sets have fixed size.

In a Release 8.0 TABLO-generated program, this Coefficient is declared in the Fortran code (the relevant module) to have this size.

In TABLO-generated programs written since this memory sharing was allowed, these Coefficients are now declared in the Fortran code as allocatable arrays. This may slightly slow down the running of TABLO-

generated programs. [We expect any slowing down to be hardly noticeable. If this is found too significant, we can easily allow users to write the Release 8.0 style TABLO-generated programs.[49] Such programs will not allow memory sharing between Coefficients and Nonzeros.]

[49] At present, DEBUG option number 172 in TABLO writes such TABLO-generated programs.

CHAPTER 5

5. LU Decomposition

The CPU time taken for the LU decomposition often dominates the CPU time taken for the rest of the simulation, especially for large models.[50]

In this chapter, we describe ways in which Release 9.0 can be quicker than Release 8.0 (see section 5.2).

There are various new ways in which you can experiment and possibly reduce considerably the CPU time taken for your model. Options you can try if your model is slow to LU decompose are:

- setting the starting values of MMNZ, MMNZ1 and MMNZ2 – see section 5.1.
- whether the memory is reallocated inside the MA48 routines or not – see section 5.2.
- whether to use compressions in MA48 – see section 5.2.1.
- whether to LU decompose the original matrix or its transpose in MA48 – see section 5.3.
- whether to use Markowitz pivot strategy in MA48 – see section 5.4.

Some basic choices available in Release 8.0 and earlier not mentioned in the list above are described in chapter 12 of GPD-3. Condensation can often reduce the time to solve a model. You should still check whether MA28 is faster than MA48 (and now MA48 transpose). You can adjust the level of accuracy for the your simulation. For example, it is much slower to use 10 subintervals if you find you get quite acceptable accuracy using 2 subintervals. However you would need to look at the accuracy summaries in the log file and the extrapolation accuracy file to check that the accuracy is still sufficiently good.

5.1 Memory for LU Decomposition in Simulations

This relates to the memory required by GEMSIM, TABLO-generated programs and SAGEM for the LU decomposition. The Release 8.0 situation is described in sections 13.3 and 13.4 of GPD-3.

There are 3 arrays in these programs which hold entries in the LU decomposition. In Release 8.0, two of these arrays have dimension MMNZ1 and one has dimension MMNZ. Now the first two may have different sizes which are called MMNZ1 and MMNZ2. The programs now report information about memory in terms of MMNZ, MMNZ1 and MMNZ2.

If you use the MA48 routines (see section 12.1.1 of GPD-3), MMNZ and MMNZ1 will be the same but MMNZ2 may be smaller.

If you use the MA28 routines (see section 12.1.1 of GPD-3), MMNZ1 and MMNZ2 will be the same but MMNZ may be smaller.

For Release 9.0 and later, there is the Command file statement

```
start with MMNZ2 = <integer value> ;
```

This statement is available in Command files for GEMSIM, TABLO-generated programs and SAGEM.

This complements the Release 8.0 statements

```
start with MMNZ|MMNZ1 = <integer value> ;
```

(see section 13.3 of GPD-3). You can set separate starting values for MMNZ, MMNZ1 and MMNZ2 using these three statements. In particular, it now makes sense to set separate starting values for MMNZ and MMNZ2 in some models when you use MA48.

---

[50] See, for example, the CPU times reported in a footnote in section 4.2.7.

Example. The GC10E1.CMF simulation with GTAP61 (see section 6.4.3 of GPD-3) requires MMNZ and MMNZ1 to be at least about 202,000 and MMNZ2 to be at least about 126,000. For this simulation it would be appropriate to include the statements

```
start with MMNZ = 210000 ;
start with MMNZ2 = 130000 ;
```

[Provided you have sufficient memory, it is usually a good idea to set these values slightly more than the minimum.]

In general, a statement of the form

```
start with MMNZ = <integer value> ;
```

sets all three values MMNZ,MMNZ1 and MMNZ2 to the specified value. You can set different values for MMNZ1 and MMNZ2 (if you wish) using the other two statements. [In the example above, we want MMNZ1 to have the same value as MMNZ so do not need a "start with MMNZ1" statement. But we include a "start with MMNZ2" statement since we want MMNZ2 to have a different value from MMNZ.]

5.1.1 Alerting About MMNZ, MMNZ1, MMNZ2 Re-Allocations

If MMNZ, MMNZ1, MMNZ2 need to be increased in order to finish the LU decomposition (see section 5.2),

- the simulation may take considerably longer than if suitable initial values for MMNZ, MMNZ1, MMNZ2 had been set via "start with MMNZ|MMNZ1|MMNZ2 = <value> ;" statements in the Command file.
- GEMSIM and TABLO-generated programs put a strong ADVICE message near the end of the run about this. [This message appears just after the minimum values for MMNZ, MMNZ1 and MMNZ2 are displayed.] Also, each time the program is doing the LU decomposition (when the message "(Calculating the LU decomposition.)" stays on the screen for a long time with a large model), the same advice is shown on the screen.

Whenever one or more of MMNZ, MMNZ1 and MMNZ2 need to be increased, you will see a report in your LOG file. [If you are using memory sharing – see section 4.2 – you will also see reports when the values of MMNZ etc are reduced.][51]

5.2 Memory Reallocation Now Done Within the MA48 Routines

There are new versions of the MA48 routines (these are used unless you explicitly ask for MA28 – see section 12.1.1 of GPD-3) which can reallocate memory (that is, increase the values of MMNZ, MMNZ1 and MMNZ2 – see section 5.1) if necessary without having to start the LU decomposition again.

In Release 8.0, if one of these values needed to be increased, the MA48 routines indicated this to the rest of the program. Then GEMSIM or the TABLO-generated program increased the sizes of the relevant arrays in memory, redid the submatrices and then called the MA48 routines to start the LU decomposition again from scratch. Having to start again from scratch can waste significant amounts of CPU time.

In Release 9.0 and later, when one of these values needs to be increased, the new MA48 routines do the increase and continue from where they left off. In particular, the submatrices are not redone and the LU decomposition is not restarted. This means that the LU decomposition will be finished significantly more quickly. Thus there is less penalty if MMNZ etc start with values which are too small. Although the penalty is less, there is still some penalty and you would be well advised to include appropriate

```
start with MMNZ|MMNZ2 = <integer value> ;
```

[51] These reports may mention the routine TGSVRAMNZ which does the reallocation.

(see section 5.1) statements in your Command file once you know the relevant values from a previous LOG file.

You can influence how fast MMNZ etc are increased if they need to be increased – see section 5.2.2 below.

In our experience, when the values of MMNZ etc are large enough, the new MA48 routines (the ones which can reallocate memory) do not take any longer than the original MA48 routines (the ones which cannot reallocate memory). However, if you want to use the original MA48 routines (and you are confident that you have set MMZ and MMNZ1 large enough), you can do so by including the statement

```
ma48 use_original = yes ;  ! the default is NO
```

in your Command file for GEMSIM and TABLO-generated programs. [This is not allowed in a Command file for SAGEM.]

The original MA48 routines are called MA48A and MA48B. The new ones have "G" (for GEMPACK) at the end of their names, which are MA48AG and MA48BG.

5.2.1 Compressions if Using New MA48A Routine

In order to minimise the size of MMNZ etc required, MA48A can do what are called **compressions**. Although doing compressions may reduce the size of MMNZ etc required, it can add significantly to the CPU time taken if a large number of compressions are required.

The original MA48A always does compressions. You can tell the new MA48A routine MA48AG (see section 5.2 above) whether or not to do compressions by including a statement of the form

```
MA48 compress = yes|NO|few ;   ! NO is usually the default – see below
```

in your Command file. This statement is available in Command files for GEMSIM, TABLO-generated programs and SAGEM.

- We assume that minimising CPU time is a high priority for most users, so the default value is usually "no" which means that no compressions are done. However, if you are sharing memory (see section 4.2), compressions are done (that is, the default is "yes") since presumably you wish to minimise the amount of memory including that for the nonzeros.
- If you include the statement "MA48 compress = yes ;", MA48AG always compresses and only increases MMNZ etc if still required after compression is done.
- If you include the statement "MA48 compress = few ;", MA48AG will do at most 5 compressions (on each step of a multi-step calculation).

Statements of the form "MA48 compress = " are only relevant if you are using the new MA48 routines. They have no effect if you are using the original MA48 routines (see section 5.2).

If compressions take a significant amount of time during a run, you will see a report indicating the total time taken for compressions near the end of the log file.

Remember that the values of MMNZ etc may need to be higher if no or few compressions are used but that the CPU time may be reduced. We give some examples below.[52]

[52] MA48 is used in all cases so the minimum MMNZ1 value is the same as the minimum MMNZ value. The results are very sensitive to a number of factors, which is why we give only approximate values in the table. In particular, the minimum values reported depend on the initial settings of MMNZ etc (possibly set via "start with mmnz" statements). They can also change if the compiler changes (LF90 or LF95) or if the compiler options change.

Effect of Compressions on Minimum MMNZ, MMNZ2 (MA48AG,MA48BG)

	No compressions ma48 compress = no ;	**Few compressions** ma48 compress = few ;	**Always compress** ma48 compress = yes ;
10x10 GTAP61 **GC10E1.CMF**	min MMNZ= 196000 min MMNZ2=174000	min MMNZ= 196000 min MMNZ2=130000	min MMNZ= 196000 min MMNZ2=130000
38x39 GTAP **1-step Euler**	min MMNZ= 34 mill min MMNZ2=29 mill	min MMNZ= 34 mill min MMNZ2=28 mill	min MMNZ= 26 mill min MMNZ2=15 mill
TERM WATER **1-step Euler**	min MMNZ= 7.8 mill min MMNZ2=6.2 mill	min MMNZ= 7.8 mill min MMNZ2=3.7 mill	min MMNZ= 7.8 mill min MMNZ2=3.7 mill
USAGE **1-step Euler**	min MMNZ = 13 mill min MMNZ2=13 mill	min MMNZ = 13 mill min MMNZ2=13mill	min MMNZ = 13 mill min MMNZ2=13mill
55x50x8 MMRF **1-step Euler**	min MMNZ = 3.4 mill min MMNZ2=3.4 mill	min MMNZ = 3.4 mill min MMNZ2=3.4 mill	min MMNZ = 3.4 mill min MMNZ2=3.4 mill
115x113 MONASH **1-step Euler**	min MMNZ = 748000 min MMNZ2=662000	min MMNZ = 748000 min MMNZ2=649000	min MMNZ = 748000 min MMNZ2=649000

5.2.2 Reallocations if Using the New MA48 Routines

You can influence how quickly the new MA48 routines increase MMNZ (if they need to) by including a statement of the form

```
MA48 increase_MMNZ = slow|medium|FAST|veryfast ;
```
! FAST is usually the default

in your Command file. This statement is available in Command files for GEMSIM, TABLO-generated programs and SAGEM.

We assume that minimising CPU time is a high priority for most users, so the default value is usually "fast". However, if you are sharing memory (see section 4.2) in GEMSIM or a TABLO-generated program, the default is "medium" since presumably you wish to be frugal with the amount of memory including that for the nonzeros.

If you include the statement "MA48 increase_MMNZ = **slow** ;" then these routines always use the minimum value that might be sufficient.

If you include a statement "MA48 increase_MMNZ = " and end with medium, fast or veryfast, these routines always attempt to use values which are larger than the minimum value that might be sufficient.

- If you include the statement "MA48 increase_MMNZ = **medium** ;" the routines will try to use values 10% higher than the minimum values that might be sufficient (if there is sufficient memory).
- If you include the statement "MA48 increase_MMNZ = **fast** ;" [this is the default] the routines will try to use values 25% higher than the minimum values that might be sufficient (if there is sufficient memory).
- If you include the statement "MA48 increase_MMNZ = **veryfast** ;" the routines will try to use values 50% higher than the minimum values that might be sufficient (if there is sufficient memory).

The danger with increasing MMNZ too quickly is that you may take more memory from the operating system than is necessary. This may slow down the program (especially if it then has to operate in virtual memory – see section 4.2.2) or other programs you are running.

On the other hand, increasing MMNZ takes some CPU time (since the program has to write the current contents of the large associated arrays to the disk and then read them back). So the less often you do it, the better.

Statements of the form "MA48 increase_MMNZ = " are only relevant if you are using the new MA48 routines. They have no effect if you are using the original MA48 routines (see section 5.2). Nor do they have any effect if you have started with values of MMNZ, MMNZ1 and MMNZ2 which are large enough.

If you need to increase MMNZ several times during a run and these reallocations take a significant amount of time during a run, you will see a report indicating the total time taken near the end of the log file.

5.2.3 Reallocations if Using the MA28 Routines

Note that the MA28 routines handle this situation much as in Release 8.0. If you use the MA28 routines and the programs needs to increase MMNZ,MMNZ1 or MMNZ2, the LU decomposition needs to be completely restarted after the extra memory is allocated. In Release 8.0, all the submatrices were recalculated in this case. Now we have reorganised the code so that redoing the submatrices is no longer needed.

5.3 LU Decomposing Transpose of LHS May be Much Quicker

Normally the LU decomposition of the Left Hand Side (LHS) Matrix is calculated at each step – see section 12.1 of GPD-3. Often this is the most time-consuming part of a simulation.

We have found that, for some models, it is much quicker to LU decompose the transpose of the LHS Matrix and then use the LU decomposition of the transpose to solve the original (not the transposed) equations at each step.

For example,

- with the GTEM model (Australian Bureau of Agricultural and Resource Economics), the LU decomposition time with MA48 for a single step fell from about 95 minutes to less than 2 minutes for the transpose.
- for a 38x39 aggregation of GTAP, the LU decomposition time with MA48 for a single step fell from over 5 minutes to less than 4 minutes for the transpose.

We give more timings in section 5.3.1 below.

Note that LU decomposing the transpose does not add any extra overheads in the rest of the simulation.

You can experiment with your model by including the statement

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

in your Command file. This statement is available in Command files for GEMSIM, TABLO-generated programs and SAGEM.

At present ***LU decompose transpose = yes ;*** only works with MA48 (not with MA28). If you are LU decomposing the transpose of the LHS Matrix, you cannot use or save an LU file.

We are grateful to Iain Duff for pointing out that working via the transpose may help in some cases. We are also grateful to Guy Jakeman who passed on the GTEM model for us to test.

5.3.1 Timings with Different Models

Here we report timings with different models. You can see that in some cases LU decomposing the transpose helps while not in others.

We know of no *a priori* way of telling whether or not LU decomposing the transpose will help with a particular model. You need to try it out.[53]

[53] It is somewhat arbitrary whether MA48 and MA28 work in row or column order but that can make a huge difference to the CPU time (as the figures in section 5.3.1 show). That is why the developers of

When you are trying this out, you also need to compare CPU times between MA48 and MA28, and you may wish to compare with and without Markowitz pivots (see section 5.4) in the MA48 tests.

In each case the timings reported below are for the LU decomposition in the first step. Note that LU decomposing the transpose does not add any extra overheads in the rest of the simulation. CPU times can vary a little from run to run, and can vary greatly between different machines. So, when reading the table below, you should concentrate on the relative values across a row and not be too concerned with the actual times.

The figures reported[54]

- are for TABLO-generated programs are with no compressions in MA48A – that is, with the statement ***`ma48 compress = no;`*** in the Command file.
- We indicate the quickest CPU time in bold.
- Notice that, for GTEM and 38x39 GTAP, MA48 with the transpose seems quickest.

Model	**MA48**	**MA48 transpose**	**MA28**	**MA28 transpose**	**MA48 Markowitz**	**MA48 Markowitz transpose**
GTEM	5502 sec	**83 sec**	158 sec	713 sec	1600 sec	453 sec
GTAP 38x39 TG-prog[55]	319 sec	**227 sec**	454 sec	1198 sec	863 sec	345 sec
GTAP 38x39 SAGEM	541 sec	265 sec	**333 sec**	957 sec	879 sec	345 sec
USAGE 503x513	64 sec	**36 sec**	88 sec	227 sec	402 sec	74 sec
TERM WATER	44 sec	160 sec	378 sec	346 sec	47 sec	**38 sec**
MONASH 115x113	**1.0 sec**	1.9 sec	2.5 sec	8.9 sec	3.9 sec	2.1 sec
MMRF 55x8	**8.6 sec**	9.9 sec	48.8 sec	25.4 sec	27.5 sec	17.5 sec

5.4 Alternative Pivot Strategy in MA48

The Harwell routines MA48 and MA28 select pivots in different ways. MA28 uses what is called the **Markowitz pivot strategy** while, by default, MA48 uses what is called the **Zlatev pivot strategy**. This may be one reason why some users prefer to use the older MA28 routines rather than the more recent MA48 routines.

One pivot must be chosen from each row of the matrix. Different pivots must be in different columns. The matrix entries in these positions must be nonzero since the reciprocal of each of these pivot entries is used in forming the LU decomposition and in solving the system of linear equations. The pivot strategy must strike a balance between maintaining sparsity and numerical accuracy.

For more about pivots and details about these two strategies, see Duff and Reid (1993).

---

the MA28 and MA48 routines (Iain Duff and colleagues) make it easy for users to work with the transpose of the original matrix.

[54] We have used different PCs for different models so you should only compare CPU times across each row, not between rows. Although LU decomposing the transpose is not supported at present with MA28 (see section 5.3), we have checked CPU times using debug option 86 with SAGEM.

[55] See section 4.2.9.

As indicated above, by default MA48 uses the Zlatev strategy. With Release 8.0, this strategy was always used with MA48. Now you can use the alternative Markowitz strategy with MA48. If you wish to do this, put the statement

```
Markowitz pivots in MA48 = yes ;      ! the default is NO
```

into your Command file. This will make MA48 behave much more similarly to MA28 in terms of the pivoting strategy used in the LU decomposition.

You can use the statement above in Command files for GEMSIM, TABLO-generated programs and SAGEM.

Duff and Reid (1993) say in section 2.1.2: "Our original intention was to offer this [the Markowitz strategy] as the default, since it is a more thorough search than Zlatev's and is what MA28 does. Unfortunately, it can be very slow [in some cases]."

We encourage users with large models to try Markowitz pivots since that strategy may result in quicker solution times in many cases (though it may be much slower in some cases).

You can see some comparative CPU figures in section 5.3.1.

CHAPTER 6

6. AnalyseGE

You can load a subtotal solution into AnalyseGE (see section 6.1). You can decompose whatever expression you have selected in a Formula or Equation (instead of decomposing the whole RHS, for example) – see section 6.2.

AnalyseGE supports Postsim statements in TAB files – see section 2.9.

See "What's New?" in the AnalyseGE Help file and follow the links there for more details of recent changes to AnalyseGE.

6.1 Subtotals

It is possible to load a subtotals result into AnalyseGE (instead of the usual cumulative solution). Use the ***File | Load Subtotal…*** menu item on the AnalyseGE form to do this. After you select the Solution file, AnalyseGE shows you the subtotals descriptions (in a box near the top right-hand side of the AnalyseGE form). You select the one you want and then click on the **Subtotal Selected** button. AnalyseGE loads the results from this subtotal. The results for all variables are those from this subtotal.

We are grateful to Hans van Meijl for suggesting this, in particular for pointing out that it can be useful to analyse subtotals results in order to better understand the whole simulation.

6.1.1 SLTOHT Can Select a Subtotal

In order to allow AnalyseGE to load a subtotal (see above), we added a new response 'u' to indicate that you just want a subtotal solution to be put onto the output Header Array or spreadsheet file. You will see this new option if you ask SLTOHT to access a Solution file which contains one or more subtotals solutions. The relevant menu of choices will look something like that shown below.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Which solutions from this Solution file do you want :
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 all the Totals solutions available  . . . . . .[T]
 (cumulative totals and 3 subtotals)
 a single Subtotal solution  . . . . . .[u]
 The single column consisting of the Cumulative totals . . .[c]
 EXIT program now  . . . . . .[e]
```

6.2 Decomposing Selected Expression

You can select an expression in a linearised equation or on the RHS of a Formula and ask AnalyseGE to decompose that expression. You do this via the "Decompose Selected Expression" popup menu item which shows when you right-click.

For an equation, you can ask for an intelligent decomposition or a complete decomposition of the expression. For a Formula, the decomposition is always the intelligent one.

If the expression you have selected contains multiplication or division, the intelligent decomposition shows the values of the separate terms when you decompose the selected expression.

Example

```
Formula A = (B + C) * D - E – (F * G / H) ;
```

Suppose that you select (B+C)*D – E. Then you will see in the decomposition separate parts showing the values of

B + C	
D	
–E	(Note that this is not +E),
(B+C)*D – E	(the whole expression).

See "What's New?" in the AnalyseGE Help file and the links there for more details and further examples.

We are grateful to Steven Rose for suggesting that such decompositions would be useful.

CHAPTER 7

7. Subtotals With Complementarity Statements

Since October 2002 when Release 8.0 was made available, we have learned some more about subtotals results in models with complementarities.

We are grateful to Martina Brockmeier and Frank van Tongeren who passed on applications in which the subtotals results calculated on the accurate run were not as expected. These examples were an important catalyst for the new features in Release 9 for subtotals results from models with complementarities. We describe these new features below.

We are also grateful to Allan Rae who provided us with an application which has shown up a number of bugs in our beta test versions of Release 9.

In running a model containing complementarity statements, two runs are carried out. First an approximate simulation using a single Euler many-step calculation is run to see what state each complementarity is at the end of the run. Secondly an accurate run is carried out after a closure swap. The purpose of these two runs is described in chapter 16 of GPD-3.

Subtotals and the Accurate Run

Subtotals calculated during the accurate complementarity run are not easy to interpret. We do not recommend that you calculate or report them. [See section 7.3 below for more details.]

Subtotals and the Approximate Run

We recommend that you calculate and report subtotals calculated during the approximate complementarity run because they are easy to interpret and give you what you expect. See section 7.2 below for more details.

Decomposition of Results

By "decomposition", we mean that you set up subtotals where each shock applied in the simulation is in just one subtotal. The sum of the subtotals should add up to exactly the total cumulative solution.

The documentation for Release 8.0 did not tell you how to decompose your results using subtotals calculated on the approximate or on the accurate run.

Details about how to use subtotals to decompose your results during the approximate run are given in section 7.2. We recommend that you use this method. Details about decomposing results during the accurate run are give in section 7.3 below but we do not recommend that you use subtotals during the accurate run when reporting your work because the subtotals are difficult to interpret and are not robust.

The main points about decomposition are:

- To complete a decomposition on the approximate run, you need to include the Newton correction variable **$del_Comp** (see section 16.7.2 of GPD-3) in one of the subtotals (preferably its own). If this variable is the only shocked variable in a subtotal, that subtotal can be interpreted as the consequences of overshooting the complementarity bounds on Euler steps. [See section 7.2 below for more details.]
- To complete a decomposition on the accurate run, you need to include in subtotals some variables introduced by the software (see section 16.7.2 of GPD-3). Even if you had known about this, there was a bug in Release 8.0 which prevented you adding these variables into subtotal statements. That bug was fixed in Release 8.0-001 (October 2003). [See section 7.3 below for more details.]

We elaborate on these points below.[56]

---

[56] This chapter updates and supersedes the document SUBTCOMP.PDF which was distributed with Release 8.0-001 (October 2003) and later versions of Release 8.

7.1 Telling the Software When to Calculate Subtotals

There are Command file statements that apply when you are solving a model containing Complementarity statements.

You can tell the software whether to calculate subtotals on the approximate run (recommended) or on the accurate run (not recommended) via the following Command file statement.

`complementarity subtotals = APPROXIMATE|accurate ;` ! default is "approximate"

This tells if subtotals are to be calculated during the approximate run or during the accurate run.

The default is APPROXIMATE unless the simulation is only doing the accurate run.

You can include the results of the approximate run as a subtotal on the Solution file via the following Command file statement.

`complementarity approx_as_subtot = first|last|yes|no ;`

This tells if the cumulative solution from the approximate run will be added as a subtotal solution.

- **`first`** or **`yes`** means that it will be added as the first subtotal (before any subtotals defined by "subtotal" statements in the Command file),
- **`last`** means that it will be added as the last subtotal (after any subtotals defined by "subtotal" statements in the Command file),
- **`no`** means that it will not be added as a subtotal.

If there are any subtotal statements in your Command file and you are calculating these subtotals on the approximate run, the default is **`first`**. Otherwise the default is **`no`**.

Example:

Run TABLO with the TAB file MOIQ.TAB and then run a simulation using the Command file MOIQ3.CMF supplied with the Release 9 Examples files.[57] Notice that MOIQ3.CMF does not contain either of the two statements above, so the default values apply. This is equivalent to having the following statements in the Command file:

```
complementarity subtotals = approximate;
complementarity approx_as_subtot = first ;
```

There are four subtotal statements in MOIQ3.CMF:

```
subtotal p_XImp_Quota = XImp_quota changes ;
subtotal p_TIMP = TIMP changes ;
subtotal $del_comp = complementarity overshooting ;
subtotal p_XImp_Quota $del_comp p_TIMP = ALL changes ;
```

If you look at MOIQ3.SL4 in ViewSOL, it has 5 subtotal columns but the first "subtotal" is the cumulative solution from the approximate run. Click on the ViewSOL Menu item ***Description*** to see that the subtotals are:

1: Cumulative solution from the approximate run.
2: XImp_quota changes
3: TIMP changes
4: complementarity overshooting
5: ALL changes

Notice also that, as expected, results in the first subtotal (cumulative solution from the approximate run) differ slightly from the cumulative solution for the accurate run (which is the true solution of the simulation).

Subtotal results 2, 3 and 4 above are a decomposition of the solution from the approximate run. Thus they add to the solution from the approximate run but do not add exactly to the solution from the accurate run. We say more about this in section 7.2.1 below.

---

[57] You will also need the data files MO.DAT and MOIQ_RAT.DAT.

7.2 Calculating Subtotals on the Approximate Run - Recommended

We believe that the natural time to calculate subtotals in on the approximate run. You are then calculating what you expect based on the closure and shocks in the Command file for the simulation. However, if you calculate subtotals on the accurate run (when the software may modify the closure and shocks[58]), we are not sure how to interpret the results – see section 7.3 below.

If you specify subtotals whose related exogenous variables cover all the shocked variables (including the Newton correction variable **$del_Comp** – see section 16.7.2 of GPD-3), the relevant subtotal results add exactly to the cumulative solution from the approximate run. That is, it is relatively easy to decompose the approximate result into several natural subtotals (provided you remember to include the Newton correction variable $del_Comp in one subtotal).

In tests we have carried out, the subtotal results are robust to variations in the number of subintervals. [That is, you get similar subtotal results for the same simulation with one, two or several subintervals.]

7.2.1 Downside – Decomposition Results do not Add Exactly to Accurate Cumulative Solution

Unfortunately there is a downside to this. The cumulative solution from the approximate run will usually be slightly different from the cumulative solution from the accurate run. And it is the latter (the cumulative solution from the accurate run) which you will want to report as the results of your application. So, if your subtotals cover all the shocks,

- the good news is that the subtotals results from the approximate run are an exact decomposition of the cumulative solution from the approximate run (and these subtotals results are what you expect from subtotals results), but
- the bad news is that these subtotals results are not an exact decomposition of the cumulative solution you wish to report, namely the cumulative solution from the accurate run.

You noticed these points when you carried out the MOIQ3.CMF simulation in section 7.1 above. We discuss this and related simulations in more detail in section 7.4 below.

If you are not reporting a decomposition, but just some interesting subtotals, there is no problem.

If you are reporting a decomposition, you might note (perhaps in a footnote) that the decomposition results only add approximately to the results reported. Or you might like to scale the subtotals results so that they do add exactly. [If so, you will need to scale them differently for each endogenous result you report.]

We will be interested to hear from users as to how they decide to handle this dilemma.

Note that this problem is not a really serious one since subtotals results are only a rough attribution of the contribution of a group of shocks within the context of a simulation. [See chapter 11 of GPD-3 and Harrison *et al* (2000) for more details. For example, the subtotals results depend on the path of the simulation (see section 11.6 of GPD-3).] In particular, you should never regard subtotals results as definitively as you regard the accurate cumulative solution from a simulation.

[58] See GPD-3 chapter 16 for the reasons why the closure and shocks are modified in a simulation when the model contains a complementarity.

7.3 Calculating Subtotals on the Accurate Run – Not Recommended

In order to calculate on the accurate run subtotals which add to the cumulative result, you need to include in subtotals some variables added by the system for complementarities.

We do not recommend reporting subtotals results calculated during the accurate run.

- We do not know how to interpret the subtotal results reported from some standard simulations.
- The results are not robust since the results calculated change significantly as the number of subintervals changes.

The results in section 7.4.4 illustrate these points.

Notice that the closure on the accurate run may change from subinterval to subinterval. This is because the target states can vary on each subinterval. [For example, there may be no state changes during subinterval 1 but one or more state changes during the second subinterval.]

For these reasons, if you calculate subtotals on the accurate run, you will see that the shocked components included in each subtotal are re-examined at the start of the accurate run in each subinterval.

7.3.1 Decomposing Accurate – Include Extra Variables in Subtotals

If, despite our recommendation in section 7.3 above, you want to produce subtotals results which decompose the accurate simulation result, you need to explicitly include the relevant extra variables (as documented in section 16.7.2 of GPD-3) in subtotals statements. These variables are

- the complementarity expression variable (whose name ends with "@E") – for example, c_IMPQUOTA@E in the MOIQ3 example below,
- the complementarity variable (p_TIMP_QUOTA in the MOIQ3 example below).

[You never need to include the lower bound or upper bound variables (their names end with "@L" or "@U" – see section 16.7.2 of GPD-3) since they are always endogenous (if they are needed).][59]

Note also that you can use levels names rather than the associated linear variables.

For example, the four subtotals statements we have included in MOIQ3AC.CMF

```
subtotal XImp_Quota = XImp_Quota changes ;
subtotal IMPQUOTA@E TIMP_QUOTA = IMPQUOTA state changes ;
subtotal TIMP = TIMP changes ;
subtotal XImp_Quota IMPQUOTA@E TIMP_QUOTA TIMP = ALL changes ;
```

use levels names.

In general, for each complementarity we recommend that you include a separate subtotal statement which reports the consequences of any state changes in that complementarity.

For example, suppose you have a Complementarity called COMP1 in the TAB file and that the associated complementarity levels variable is called LEV_COMP1. Then you could include the subtotal statement

```
subtotal  LEV_COMP1 COMP1@E = COMP1 state changes ;
```

[59] In fact the subtotals results reported still do not decompose the simulation results reported for variable $del_Comp. That variable is shocked by 1 in the accurate run, but all subtotals results for it will be zero. That variable plays no role in the accurate run since it only occurs in the non-differentiable equations which are turned off in the accurate run. To get a decomposition of the results for this variable, we should not shock it during the accurate run. However that would cause otherwise unnecessary complications in the code so we have chosen to leave this apparent subtotals anomaly relating to variable $del_Comp.

to capture the consequences of any state changes in that complementarity. [You would probably want to include such a subtotal statement for every complementarity in the model.]

In the example MOIQ3AC.CMF, the subtotal is called IMPQUOTA and the associated complementarity variable is TIMP_QUOTA, so the subtotal statement is (using levels names)

```
subtotal IMPQUOTA@E TIMP_QUOTA = IMPQUOTA state changes ;
```

or, using linear variable names,

```
subtotal c_IMPQUOTA@E p_TIMP_QUOTA = IMPQUOTA state changes ;
```

See the MOIQ3AC.CMF examples given in section 7.4.1 below.

7.4 Examples: MOIQ3.CMF and Related Simulations

In Release 8.0 of GEMPACK, the examples subdirectory contains the TABLO Input file for the MOIQ model, MOIQ.TAB, and also the Command file MOIQ1C.CMF which runs a simulation to decrease the import quota volume of commodity 1 by 30%. This decrease makes the import quota binding for commodity 1. The shock in the Command file MOIQ1C.CMF is

```
shock p_XImp_Quota = -30  0 ;
```

Below we use a simulation in MOIQ3.CMF that is similar to the Command file MOIQ1C.CMF distributed with Release 8.0 of GEMPACK.[60]

The difference between the original simulation in MOIQ1C.CMF and the simulation in MOIQ3.CMF is that an additional shock has been added. The additional shock statement in MOIQ3.CMF is

```
shock p_TIMP = uniform 5 ;
```

This extra shock is added in this example so that we have more shocks to use in examples of subtotal statements.

The closure and shocks in MOIQ3.CMF are shown below.

Closure and Shocks in MOIQ3.CMF

```
exogenous p_PIMP p_FEXP p_XEXP("c2") p_PHI
          p_V 1 p_FWAGE p_CR p_TIMP p_XCAP
          p_XImp_Quota
   ;
rest endogenous ;
shock p_XImp_Quota = -30  0 ;
shock p_TIMP = uniform 5 ;
```

In MOIQ1C.CMF and in MOIQ3.CMF, there are 20 Euler steps on the approximate run.

```
complementarity steps_approx_run = 20 ;
```

[60] The shocks in MOIQ3.CMF are the same as in the Command file MOIQ2.CMF we distributed with Release 8.0-001 (October 2003). However MOIQ3.CMF and its variants (which are discussed below) are set up to illustrate the Release 9 features (whereas MOIQ2.CMF and its variants were set up to illustrate the Release 8.0-001 features).

The subtotals statements in MOIQ3.CMF are shown below.

Subtotal Statements in Command file MOIQ3.CMF

```
! Subtotals – decompose cumulative result from approximate run
subtotal p_XImp_Quota = XImp_quota changes ;
subtotal p_TIMP = TIMP changes ;
subtotal $del_comp = complementarity overshooting ;
subtotal p_XImp_Quota  $del_comp  p_TIMP = ALL changes ;
```

As discussed in section 7.2 above, in order to get a decomposition on the approximate run, you need to include the Newton-correction complementarity variable **$del_Comp** in one subtotal. The variable $del_Comp is used during the approximate run to generate Newton correction terms that force the solution path back onto the appropriate branch of the complementarity condition if the step has overshot. This is discussed in section 16.7.2 of GPD-3 and in section 4.14 of GPD-2.

You must not add $del_Comp to the list of exogenous variables, or to the list of shocks, in your Command file – it is automatically exogenous and given a shock of 1 in the approximate run (see section 16.7.2 of GPD-3). However you do need to add it to one of the subtotals if you are trying to decompose the cumulative results from the approximate run (which is what we recommend for decompositions). You can see that $del_Comp is in the third subtotal above. You will see later, when we discuss the results, why we give the description "complementarity overshooting" to this subtotal.

The simulation MOIQ3.CMF decomposes the cumulative results for the approximate run into three subtotals (the first three above) whose results add to the cumulative result. [The results of the fourth subtotal "ALL changes" above should equal the cumulative result from the approximate run since all shocked variables are in that subtotal.]

The full MOIQ3.CMF Command file is shown on the next page.

We have also supplied some variants of MOIQ3.CMF to illustrate various points. These variants, whose names begin with MOIQ3 (for example, MOIQ3AC.CMF and MOIQ32I.CMF), are discussed below. The variants have the same closure and shocks as in MOIQ3.CMF but have different Command file statements to say when subtotals are calculated and to vary the number of subintervals.[61]

[61] The MOIQ3 examples here update and supersede the MOIQ2 examples which were distributed with Release 8.0-001 (October 2003) and later versions of Release 8. We do not recommend running the MOIQ2 examples with Release 9 since the MOIQ3 examples discussed in the text illustrate the salient features better.

Command file MOIQ3.CMF

```
! GEMPACK Command file for running
!  GEMSIM or the TABLO-generated program
!  for the Mixed Levels/Linear version  (MOIQ.TAB)
!  of the  Miniature ORANI model (not condensed) with Import Quotas

! Steps in approximate run
complementarity steps_approx_run = 20 ;

! Solution method and steps
method = gragg ;
steps = 6 8 10 ;

verbal description =
Quota test.
Illustrate decomposition of results from approximate run ;

!   Auxiliary files
auxiliary files = moiq ;

! Input data
file basedata  = mo.dat ;
file quota_ratios = moiq_rat.dat ;

! Updated data
updated file basedata  = <cmf>.upd ;
updated file quota_ratios = <cmf>q.upd ;

extrapolation accuracy file = yes ;
log file = yes ;

! Closure
exogenous p_PIMP p_FEXP p_XEXP("c2") p_PHI
          p_V 1 p_FWAGE p_CR p_TIMP p_XCAP
          p_XImp_Quota  ;
rest endogenous ;

! Shocks
shock p_XImp_Quota = -30 0 ;
shock p_TIMP = uniform 5 ;

! Subtotals – decompose cumulative result from approximate run
subtotal p_XImp_Quota = XImp_quota changes ;
subtotal p_TIMP = TIMP changes ;
subtotal $del_comp = complementarity overshooting ;
subtotal p_XImp_Quota $del_comp p_TIMP = ALL changes ;
```

7.4.1 Example: MOIQ3AC.CMF Simulation

Before we discuss the results of the MOIQ3.CMF simulation, we describe a similar Command file MOIQ3AC.CMF. This carries out the same simulation but calculates subtotals which decomposes the cumulative result from the accurate simulation (rather than the approximate one as in MOIQ3.CMF).

The relevant statements in MOIQ3AC.CMF are shown below.

Subtotal Statements in Command file MOIQ3AC.CMF

```
! Subtotals for accurate run (not recommended)
! Next tells program to calculate subtotals on accurate run.
! [The default is to calculate subtotals on the approximate run.]
complementarity subtotals = accurate ;

! Subtotals - decompose cumulative result from accurate run
subtotal p_XImp_Quota = XImp_quota changes ;
subtotal p_TIMP = TIMP changes ;
subtotal IMPQUOTA@E TIMP_QUOTA = IMPQUOTA state changes ;
subtotal  XImp_Quota
          IMPQUOTA@E TIMP_QUOTA
          p_TIMP = ALL changes ;
```

The versions of GEMSIM and TABLO-generated programs for Release 8.0-001 (October 2003) and later allow you to include the complementarity variables (such as p_IMPQUOTA@E above) in subtotal statements.

The variables listed in the exogenous statement in MOIQ3.CMF are exogenous during both the approximate and the accurate runs.[62]

Also relevant to the exogenous variables for the accurate run are the complementarity variable and the complementarity expression variable (see section 16.7.2 of GPD-3). If the complementarity is in state 1 or 3 after the approximate run, the complementarity variable (p_TIMP_QUOTA in the MOIQ model) is exogenous and shocked in the accurate run. If the complementarity is in state 2 after the approximate run, the complementarity expression variable (c_IMPQUOTA@E in the MOIQ model) is exogenous and shocked in the accurate run.

To include the effect of all shocks in the accurate run, you need the third subtotal above

```
subtotal  c_IMPQUOTA@E  p_TIMP_QUOTA = IMPQUOTA state changes ;
```

The description (IMPQUOTA state changes) suggests that you might interpret the results of this subtotal as showing the effects of state changes. You will learn more about that when we discuss the results in section 7.4.4 below.

7.4.2 Several Subintervals

When discussing the results, it is useful to also have results from this simulation solved over several subintervals (see section 7.3 of GPD-3). We have supplied the following Command files.

- **MOIQ3.CMF** – the basic simulation (see section 7.4). The subtotals here decompose the cumulative results from the approximate run. Just one subinterval.
- **MOIQ3AC.CMF** – the same simulation but subtotals decompose the cumulative result from the accurate run (see section 7.4.1). Just one subinterval.

[62] Note that $del_Comp is not included in the list of exogenous variables in the Command file. This variable is exogenous for the approximate run but endogenous for the accurate run.

- **MOIQ32I.CMF** and **MOIQ39I.CMF** – same as MOIQ3.CMF except these are run over 2 and 9 subintervals respectively.
- **MOIQ3AC2.CMF** – same as MOIQ3AC.CMF except there are 2 subintervals.

If you have access to Release 9 of GEMPACK, we suggest that you run these 5 simulations now so that you can look at the results as you read our discussion of the results in section 7.4.3 below.

7.4.3 MOIQ3.CMF Subtotals Calculated During Approximate Run

In the following sections we will look just at one variable p_CHOUS but the other variables behave in a similar way. You will find the variable p_CHOUS under the heading Macros in ViewSOL.

In MOIQ3.SL4 produced by MOIQ3.CMF, the p_CHOUS result reported (this is from the accurate version of the simulation in each case) is 10.755.

p_CHOUS = 10.755 (accurate simulation result from MOIQ3.CMF)

In MOIQ3.SL4 you can also see the cumulative result for p_CHOUS from the approximate run since the first subtotal results are the cumulative solution from the approximate run. [This is explained in section 7.1 above since the default is "`complementarity approx_as_subtot = first ;`".] To check this, after you have loaded the MOIQ3 results into ViewSOL make sure that you have selected the **moiq3** solution (in the "Choose which solution to view" drop-down box), and then click on Description in the main menu. You will see the descriptions for the different subtotals. Although there are only 4 subtotal statements in MOIQ3.CMF, the software has added the cumulative solution from the approximate run as an extra one and made it the first subtotal. Note that the second subtotal result shown from MOIQ3 shows the effects of the Ximp_Quota changes, the third shows the effects of the TIMP shocks, the fourth show the effects of the $del_Comp shock and the fifth shows the effects of all shocks. Of course the fifth subtotal results are identical to the first subtotal results.]

The p_CHOUS results from MOIQ3.CMF are

Subtotals during Approximate Run

MOIQ3.CMF	**p_CHOUS** =	10.755	(accurate cumulative result)
		10.403	(subtotal 1 – cumulative results from approximate run)
		6.435	(subtotal 2 – XImp_Quota changes)
		3.967	(subtotal 3 – TImp changes)
		0.0001	(subtotal 4 – complementarity overshooting)
		10.403	(subtotal 5 – ALL changes – same as subtotal 1)

Note that overshooting occurs since the complementarity for commodity C1 changes state after about 10% of the 15th Euler step of 20. But the $del_Comp overshooting subtotal result is very small (about 0.00001) since the software redoes that step to minimise overshooting.[63]

It is also interesting to note the p_CHOUS result for the accurate version of this simulation, which is 10.755, is not a lot different from the approximate result of 10.403.

As indicated at the beginning of this chapter, we believe that these subtotals results, calculated during the approximate run, are the natural ones to calculate and report.

As we indicated in section 7.2.1 above, there is a downside to reporting subtotals results from the approximate simulation. Even when (as is the case here) the subtotals are a decomposition of all the shocks, they do not add exactly to the cumulative result from the accurate run (though they do add exactly to the cumulative result from the approximate run). Since the p_CHOUS results (and those for other variables) are not much different between the accurate and approximate runs, this is not much of a problem.

---

[63] If you included the statement
`complementarity redo_steps = no ;`
(see section 16.6 of GPD-3) in the Command file, you would find that the $del_Comp subtotal 4 result for p_CHOUS is much larger – about 0.84.

You might also like to look at the p_CHOUS results from the Command files MOIQ32I.CMF and MOIQ39I.CMF where 2 and 9 subintervals have been used. The results from MOIQ32I.CMF are

Subtotals during Approximate Run (2 subintervals)

MOIQ32I.CMF p_CHOUS =	10.755	(accurate cumulative result)
	10.650	(subtotal 1 – cumulative results from approximate run)
	6.684	(subtotal 2 – XImp_Quota changes)
	3.966	(subtotal 3 – TImp changes)
	0.0001	(subtotal 4 – complementarity overshooting)
	10.650	(subtotal 5 – ALL changes – same as subtotal 1)

while the results from MOIQ39I.CMF are

Subtotals during Approximate Run (9 subintervals)

MOIQ39I.CMF p_CHOUS =	10.755	(accurate cumulative result)
	10.739	(subtotal 1 – cumulative results from approximate run)
	6.770	(subtotal 2 – XImp_Quota changes)
	3.969	(subtotal 3 – TImp changes)
	0.0001	(subtotal 4 – complementarity overshooting)
	10.739	(subtotal 5 – ALL changes – same as subtotal 1)

Notice that, as expected, the cumulative solution from the approximate run gets closer to the cumulative result from the accurate run when more subintervals are used.

Notice also that the subtotal results don't change by very much. This is in contrast to the subtotal results calculated during the accurate run which (as you will see in section 7.4.4 below) change dramatically if the number of subintervals is changed.

Note that there is only one variable $del_Comp introduced by the software, even if you have several complementarity statements in your model. The subtotal relating to this variable (subtotal 4 in the example above) captures overshooting for all complementarities. There are not separate $del_Comp variables which allow you to separate the effects of overshooting on different Complementarities.

7.4.4 MOIQ3AC.CMF Subtotals Calculated During Accurate Run (Not Recommended)

In MOIQ3AC.SL4 produced by MOIQ3AC.CMF, the p_CHOUS result reported (this is from the accurate version of the simulation in each case) is 10.755.

p_CHOUS = 10.755 (accurate simulation result from MOIQ3AC.CMF)

Of course, this is the same as the accurate simulation result from MOIQ3.CMF (see section 7.4.3 above).

The subtotals results in MOIQ3AC.SL4 are

- Ximp_Quota changes (effects of shocks to p_Ximp_Quota). This is subtotal number 1.
- TIMP changes (effects of shocks to p_TIMP). This is subtotal number 2.
- IMPQUOTA state changes (effects of system-supplied shocks to IMPQUOTA@E and TIMP_QUOTA). This is subtotal number 3.
- effect of all shocks. This is subtotal number 4.

Notice that the cumulative solution from the approximate run is not added automatically as a subtotal – see section 7.1 – since the default is "complementarity approx_as_subtot = no ;" when subtotals are calculated during the accurate run.

The p_CHOUS results from MOIQ3AC.CMF are

Subtotals during Accurate Run (not recommended)

MOIQ3AC.CMF p_CHOUS =	10.755	(accurate cumulative result)
	19.191	(subtotal 1 – XImp_Quota changes)
	3.485	(subtotal 2 – TImp changes)
	–11.920	(subtotal 3 – effect of state changes)
	10.755	(subtotal 4 – ALL changes – same as cumulative result)

Notice that the result for subtotal 1 (Ximp_Quota changes) is dramatically different from MOIQ3.CMF which calculates the subtotals on the approximate run. However the subtotal 2 results (TImp changes) are not so different.

We cannot find any natural interpretation for the first and third subtotals above (the Ximp_Quota changes and the IMPQUOTA state changes results). Since these results are not numerically robust (see just below) perhaps this is not surprising. Subtotals usually depend heavily on the closure and the path so that it is very difficult to interpret the results if the closure and path are changed through the simulation, as happens in the accurate run.

The p_CHOUS results from MOIQ3AC2.CMF (same as MOIQ3AC.CMF except use 2 subintervals) are

Subtotals during Accurate Run 2 subintervals (not recommended)

MOIQ3AC2.CMF p_CHOUS =	10.755	(accurate cumulative result)
	10.656	(subtotal 1 – XImp_Quota changes)
	3.841	(subtotal 2 – TImp changes)
	–3.742	(subtotal 3 – effect of state changes)
	10.755	(subtotal 4 – ALL changes – same as cumulative result)

Notice that the results for subtotals 1 and 3 are dramatically different from those from MOIQ3AC.CMF when only one subinterval is used. This is a clear illustration as to why we do not recommend calculating subtotals during the accurate run.

- Subtotals results calculated during the accurate run are not robust. [For example, they can change a lot if the number of subintervals is changed.]
- Hence we do not know how to interpret subtotals results when calculated during the accurate run.

We elaborate on this in section 7.4.5 below.

Fortunately, these problems do not occur when subtotals are calculated during the approximate run.

7.4.5 Subtotals from Accurate Run Not Robust

The subtotals results from the accurate run change dramatically if several subintervals are used. We report some of these results (based on MOIQ3AC.CMF with one or more subintervals) below.

We think we understand why these results are not robust and include our reasons later in this section.

The results below come from calculating subtotals in MOIQ3AC.CMF on the accurate run.

- Subtotal 1 refers to the XImp_Quota changes,
- Subtotal 2 refers to the TIMP changes, and
- Subtotal 3 refers to the IMPQUOTA state changes.

The p_CHOUS results from MOIQ3AC.CMF are over 1 subinterval (for 6 8 10 Gragg)

10.755 (sim) 19.191 (subtot 1) 3.485 (subtot 2) -11.920 (subtot 3)

Now, suppose that we solve this over 2 subintervals (6 8 10 Gragg) [MOIQ3AC2.CMF]. The p_CHOUS results are

10.755 (sim) 10.657 (subtot 1) 3.841 (subtot 2) -3.742 (subtot 3)

Note that subtotals results 1 and 2 have changed dramatically.

Now, suppose that we solve this over 20 subintervals (6 8 10 Gragg). The p_CHOUS results are

10.755 (sim) 7.668 (subtot 1) 3.934 (subtot 2) -0.847 (subtot 3)

Note that subtotals results 1 and 2 have changed dramatically again.

We think that we understand what is happening, at least in this simulation. We explain this below. This explanation relies on a detailed understanding of the closure swap made during the accurate simulation, and is somewhat complicated. It is not essential that you understand this explanation, so feel free to skip it.

> In this simulation, there is just one state change. The state of complementarity IMPQUOTA for the first commodity C1 changes from being in state 1 (non-binding quota) to state 2 (binding quota) about three-quarters of the way along the simulation. When we break the simulation up into 20 subintervals, this state change happens in just one of them (actually in subinterval number 14). It is only in this subinterval that the complementarity variables IMPQUOTA@E and TIMP_QUOTA which make up subtotal number 3 receive nonzero shocks. [To understand this, see section 16.7.1 of GPD-3.] So subtotal number 3 only has a nonzero value in this single subinterval. And, the size of the subtotal in that single subinterval cannot be too large because only part of the overall shock is being applied there.
>
> Indeed, suppose that we broke up the simulation into just three subintervals of unequal length. Suppose that the first subinterval takes the simulation up until just before this state change. Suppose that the second subinterval is very small and just takes the simulation from where the quota is nearly binding to where it has just become binding. And suppose that the third subinterval completes the simulation. Clearly the subtotal 3 result will be very small since it only has a nonzero value in the tiny subinterval number 2.
>
> This is the intuition behind our understanding of subtotal number 3. If you use a very large number of equal-sized subintervals, the result for subtotal number 3 must be small since it only gets a nonzero value in one of these subintervals. As the number of subintervals increases, this subtotal number 3 must get closer and closer to zero.
>
> To check that, we ran the simulation with 100 subintervals. The results for p_CHOUS are
>
> **10.755 (sim) 6.884 (subtot 1) 3.936 (subtot 2) -0.094 (subtot 3)**
>
> Sure enough, the subtotal 3 result has become smaller again, as expected.
>
> Notice also that the results for subtotal number 2 (effects of the TIMP shock) are hardly changing as we vary the number of subintervals. This subtotal result is almost the same as the corresponding subtotal calculated during the approximate run. Indeed the subtotals results for p_CHOUS calculated during the approximate run are (see the MOIQ3.CMF results quoted in section 7.4.3 above)
>
> **10.403 (subtot 1) 6.435 (subtot 2) 3.967 (subtot 3) 0.0001 (subtot 4)**
>
> Here subtotal number 1 is the cumulative solution from the approximate run, subtotal number 2 is the effect of the XIMP_QUOTA shock, subtotal number 3 is the effects of the TIMP shock and subtotal number 4 is the effects of the $del_Comp shock. Thus the effects of the TIMP shock are very similar in the approximate run and in the accurate run, irrespective of the number of subintervals used for the accurate run. And the effects of the XIMP_QUOTA shock are very similar in the approximate run (6.435) and in the accurate run when many subintervals are used (6.884 with 100 subintervals).
>
> This provides another perspective on calculating subtotals on the accurate run. We don't trust them since they are not robust (they change greatly if we change the number of subintervals), as the example above shows. Perhaps they give meaningful results if we were to use a very large number of subintervals since that limits the influence of the shocks to the quota variables (IMPQUOTA@E and TIMP_QUOTA in this model). But there is no point in having to use a large number of subintervals to calculate subtotals in the accurate case. You can more easily calculate subtotals during the approximate run (usually with just one subinterval) and subtotals calculated during the approximate run seem to capture what we expect of subtotals.

This completes our example which shows conclusively that subtotals results on the accurate run are not robust because they may change dramatically if you use more or less subintervals.

7.4.6 Subtotals from Approximate Run Are Robust

Contrast the non-robustness of subtotals from the accurate run with subtotals calculated on the approximate run. We have seen in section 7.4.3 above that, when a decomposition of the approximate simulation is calculated using 20 Euler steps on 1, 2 or 9 subintervals, the results hardly change as we vary the number of subintervals.

Below

- Subtotal 1 is the cumulative solution from the approximate run,
- Subtotal 2 refers to the XImp_Quota changes,
- Subtotal 3 refers to the TIMP changes, and
- Subtotal 4 refers to the effects of overshooting ($del_Comp shock).

For 20 Euler steps with 1 subinterval, the p_CHOUS values are:

10.370 (subtot 1) 6.435 (subtot 2) 3.967 (subtot 3) 0.00001 (subtot 4)

For 100 Euler steps with 1 subinterval [MOIQ3A2.CMF], the p_CHOUS values are:

10.683 (subtot 1) 6.715 (subtot 2) 3.969 (subtot 3) 0.00002 (subtot 4)

The cumulative result from this approximate run has moved closer to the accurate result of 10.755. The other subtotal results have not changed significantly.

For 1000 Euler steps with 1 subinterval [MOIQ3A3.CMF], the p_CHOUS values are:

10.748 (subtot 1) 6.779 (subtot 2) 3.969 (subtot 3) 0.000002 (subtot 4)

This further reinforces our recommendation to calculate and report subtotals from the approximate run, rather than from the accurate run.

Subtotals results from the approximate run are robust and relatively easy to interpret.

However,

Subtotals results from the accurate run are not robust and are difficult to interpret.

7.4.7 The Initial Closure is Shown on Solution Files

Although the cumulative results shown on the Solution file (and hence, in ViewSOL and AnalyseGE) are those from the accurate simulation, the closure stored on the Solution file is the original closure (that is, the closure for the approximate simulation). [Remember that ViewSOL shows results for exogenous variables in red and results for endogenous variables in black.] We think that the closure from the approximate simulation is the natural one for you to see since it is the closure you specified in the Command file.

We apologise for not documenting this in the Release 8.0 documentation.

For example, in the MOIQ3.CMF simulation in section 7.4 above, ViewSOL indicates that both components of the variable p_IMPQUOTA@E are endogenous, even though the C1 component is shocked by –0.2 in the accurate simulation, and is included in the third subtotal in section 7.4.1 above. Similarly ViewSOL shows that the both components of the complementarity dummy variable $IMPQUOTA@D are exogenous even though they are endogenous in the accurate simulation (see section 16.7.2 of GPD-3).

7.5 Variables with No Components Exogenous Allowed in Subtotals

In order to allow subtotals which constitute a decomposition to be calculated during the accurate run, we softened [in Release 8.0-001 (October 2003)] the testing of variables that are allowed in a subtotal statement. You are now allowed to include all components of a variable, none of whose components

are exogenous. [But you are not allowed to specify just some components of a variable if these components are not exogenous.][64]

Example. Consider the third subtotal statement in MOIQ3AC.CMF (see section 7.4.1 above).

```
subtotal  IMPQUOTA@E TIMP_QUOTA = IMPQUOTA state changes ;
```

Here all components of variables IMPQUOTA@E and TIMP_QUOTA are included in this subtotal.

If there are no state changes then, since both quotas are non-binding in the initial data, both components of TIMP_QUOTA will be exogenous (and given shocks of zero) while no components of IMPQUOTA@E will be exogenous. Under the checking rules in force in Release 8.0, that statement would not be allowed. It is now allowed.

[64] When you specify the variables and components summed over in a subtotal statement on a Command file, if the Command file is for GEMSIM or a TABLO-generated program, you are choosing a subset of the exogenous variables. But if the Command file is for SAGEM, you are choosing a subset of the shocked variables.

CHAPTER 8

8. Enhancements to the TABLO Language

Here we document various enhancements to the TABLO language for Release 9.0 and later. We also include some points of clarification about the Release 8 documentation in GPD-2 – see section 8.7 below.

There are other enhancements to the TABLO language described elsewhere in this document. See, for example, sections 2.5, 3.2, 3.3.1 and 9.1.

8.1 Reads Only if Header Exists – Read Qualifier IfHeaderExists

You can use the Read qualifier **`IfHeaderExists`** to ensure that a read statement is carried out only if the specified header exists on the file. If the header is present, the data is read from it. Otherwise no error occurs. This may be useful in a number of circumstances.

Example 1.

> You may have a general-purpose model which often runs with default elasticity values of 1 (or zero). But, in some cases you may want to be able to provide more specific elasticity values. The same TAB file can be used for the two cases if you set the elasticity values to their default value and then read in the more explicit values only if the relevant header exists on the input data file. The TAB file statements might be as follows.
>
> ```
> Coefficient (All,c,COM) ELAST(c) ;
> Formula (All,c,COM) ELAST(c) = 1 ;  ! the default value !
> Read (IfHeaderExists) ELAST from File INPUT Header "ELAS" ;
> ```
>
> The "IfHeaderExists" is a Read qualifier allowed for Release 9.0 and later. The read is carried out only when the relevant header exists on the Header Array file. If the header is there, the values are read from it. Otherwise the values already set up are used.

Example 2.

> Consider a recursive dynamic model in which you wish to keep track of price changes across a number of years. It is often ok to consider all basic prices being equal to one in the first year of a sequence of linked simulations.
>
> You can avoid having to add lots of price headers (with all values equal to 1) to the data for the first year if you use Read(IfHeaderExists) to read the prices. For each relevant price, put a Formula setting the prices to 1 and then put a Read(IfHeaderExists) statement after. In the simulation for the first year, the header will not exist so all prices will be set to one, but the updated prices will be written to the updated data file (even though the header is not present on the input data – see section 8.1.1 below). In year 2, the updated prices from the simulation for year 1 will be on the updated data from that year, and these price values will be read at the start of year 2. Similarly for subsequent years.

8.1.1 Syntax and Semantic Rules

- **`IfHeaderExists`** is a Read qualifier. It is only allowed in Read statements in which a Header is specified.
- The read is only carried out if the specified header is on the file. [For this reason, you must be careful to set up alternative values that will be used if the header is not present. You must set these values up in statements that come earlier in the TAB file.] If the header is not present, no error occurs.

- If the file from which the Read(IfHeaderExists) is made is updated, the relevant header will always be put on the updated data (irrespective of whether or not the header was present in the starting data file read).

8.2 Default Statements for Bounds on Coefficients in TAB Files

Coefficient and levels Variable declaration statements can have qualifiers that specify acceptable ranges of values for the coefficient and levels variable (see sections 3.3 and 3.4 of GPD-2). For example,

```
COEFFICIENT (GE 0.0) (All,c,COM)  V4BAS(c) ;
VARIABLE (LEVELS, > 0.0, < 10.0) (All,c,COM)  V5BAS(c) ;
```

There are now two DEFAULT statements which allow you to set the range restrictions for many COEFFICIENTs at once.

COEFFICIENT (DEFAULT=LOWER_BOUND <condition>) ;

COEFFICIENT (DEFAULT=UPPER_BOUND <condition>) ;

For the LOWER_BOUND, the conditions allowed are:

GT real-number or GE real-number, eg GT 0.0 or >= -1.0

For the UPPER_BOUND, the conditions allowed are:

LT real-number or LE real-number, eg LT 100.0 or <= 1.0

These defaults can be turned off by using:

COEFFICIENT (DEFAULT=LOWER_BOUND OFF) ;

COEFFICIENT (DEFAULT=UPPER_BOUND OFF) ;

The default statements also apply to the Coefficients defined by a levels variable declaration.

Example.

```
COEFFICIENT (DEFAULT=LOWER_BOUND   GT 0.0) ;
COEFFICIENT  X ;
(all,i,IND) Y(i);
COEFFICIENT (GT  -1.0) Z ;
VARIABLE(LEVELS) (all,i,IND) V(i) ;
COEFFICIENT (DEFAULT=LOWER_BOUND OFF) ;
COEFFICIENT W ;
```

Here Coefficients X, Y and Coefficient V (defined via the levels variable declaration) have the range restriction "GT 0.0".
Z has its own LOWER_BOUND range restriction of "GT –1.0"
W has no LOWER_BOUND (or UPPER_BOUND) range restriction.

The two range restrictions UPPER_BOUND and LOWER_BOUND are independent of one another in the sense that they must be turned on and off separately.

8.3 Reserved Words Can Be Used As Set Element Names in TAB Files

In Release 8.0 (and earlier), reserved words (for example, PROD, SUM, AND, LE – see section 4.2.1 of GPD-2) could not be used as set element names. For example, with Release 8.0 TABLO,

```
SET SET1 (food, prod, cars) ;    ! Allowed in Rel 9 !
```

would generate the (unhelpful) semantic error message "Expecting a set element name" when the word "prod" was processed.

For Release 9.0 and later, such reserved words can be used as set element names, both inside set declarations (as above) and inside quotes in Formulas and Equations as in, for example,

```
Formula  SALES("prod") = 6 ;
```

8.4 BY_ELEMENTS Qualifier and Set Mappings

8.4.1 BY_ELEMENTS Qualifier in Formulas for Set Mappings

For Release 9.0 and later, the qualifier BY_ELEMENTS is no longer required in formulas with a set mapping on the left-hand side and an element or index-expression on the right-hand side.[65]

Example. [See Example 1 in section 4.8.11 of GPD-2]

```
MAPPING MAR2AGGMAR from MAR to AGGMAR ;
Formula (BY_ELEMENTS) (all,m,MAR) MAR2AGGMAR(m) = COM2AGGCOM(m) ;
```

In this example, the qualifier (BY_ELEMENTS) is no longer required (though TABLO does not object if it is present). That is, the statement

```
Formula (all,m,MAR) MAR2AGGMAR(m) = COM2AGGCOM(m) ;
```

is now allowed.

This is a change from the documentation in sections 3.13 and 4.8.11 of GPD-2.

TABLO will however report an error if you include the qualifier BY_ELEMENTS when the right-hand side is not an element or an index-expression. For example

```
Formula (BY_ELEMENTS) (all,m,MAR) MAR2AGGMAR(m) = 1 ; !WRONG!
```

will generate a TABLO error during the CHECK stage.

8.4.2 Writing Set Mappings BY_ELEMENTS

In Release 8.0, you could only write BY_ELEMENTS all of a set mapping – see section 4.8.13 of GPD-2. For Release 9.0 and later, you are allowed to write part of a set mapping BY_ELEMENTS. That is, the write statement can include quantifiers or can mention an element.

For example, suppose that COM2IND maps the set COM to the set IND. Then the following are allowed.

```
Write (BY_ELEMENTS)  com2ind  to file … ;
Write (BY_ELEMENTS)  (all,c,com1) com2ind(c)  to file … ; ! COM1 is a subset of COM !
Write (BY_ELEMENTS)   com2ind("c7")  to file … ;  ! if element c7 is in set COM !
```

8.4.3 Updating Set Mappings Read BY_ELEMENTS

Consider a set mapping whose values are read BY_ELEMENTS (see section 4.8.1 of GPD-2). Then there is an array of character strings on the relevant place on the file. Suppose that some other data on this file gets updated. Then an updated version of the whole file (including the array from which the values of the set mapping were read) is produced.

It was a bug in Release 8.0 that the updated data file contained integer (rather than character) data in the relevant place. This bug is fixed in Release 9.

Example

```
Read (BY_ELEMENTS) ComToInd from file basedata header "C2I" ;
```

Now there will be character data at header "C2I" on the updated version of the basedata file.[66]

---

[65] Index-expressions include elements, indices and expressions beginning with a set mapping. See section 4.2.3 of GPD-2 for a formal definition.

[66] If the read shown in this example is followed by other reads and/or formulas which complete the setting of the values of COMTOIND, the values at header "C2I" on the updated version of file basedata may be different from those at the same header in the pre-simulation basedata file. The values on the updated data will be the values of COMTOIND(c) after all reads and/or formulas for COMTOIND have been carried out. For example, there may be the formula

8.5 Seeing the Values of Integer Coefficients of Dimension 3 or Higher

Although you are allowed to have integer Coefficients of dimension 3 of higher in TAB files, it has been difficult (almost impossible) to see the values of these Coefficients. We are grateful to Robert McDougall for pointing this out and for encouraging us to give options for seeing the values of these Coefficients and the associated element names.

For Release 9.0 or later,

- any Display statement will work for Integer Coefficients of dimension 3 or higher. This includes simple Display statements such as

  `Display COEF1 ;` ! quantifiers and elements are also allowed in this case !

  or complicated ones involving quantifiers, elements and or subsets. You will see the values displayed as integers, as you would expect. You can see the element names associated with each entry of the array since Display output is labelled output – see the example output on the first page in chapter 5 of GPD-4.[67]

- you can include simple Write statements in your TAB file to write all the values of an integer Coefficient with 3 or more dimensions to a Header Array file or to a text file or to the terminal. These simple write statements must be of the form

  `Write COEF1 to file … ;` ! no quantifiers or elements are allowed !

  In each case, the output values will be written not as integers, but as real numbers. [This is because we do not support 3 or higher dimensional integer arrays on Header Array files or on GEMPACK text data files.]

- AnalyseGE can show the values of integer Coefficients of dimension 3 or more.[68]

8.6 Names for Homotopy Variables Added to Levels Equations

As documented in sections 3.9 and 3.20 of GPD-2, you can ask TABLO to add Homotopy variables to some or all Levels Equations using the ADD_HOMOTOPY qualifier in Equation or Default statements. You can see examples using this qualifier in section 7.6 of GPD-2 and section 7.6 of GPD-3. See also sections 7.6.4 and 7.6.5 of GPD-3 for documentation about this qualifier in Release 8.0.

In Release 8.0, the same Homotopy variable is added to all relevant equations, and the Homotopy Levels variable is always called HOMOTOPY (see section 7.6.4 of GPD-3).

In Release 9.0 and later, you can choose a different name for these Levels Homotopy variables and you can use different names for different equations if you wish to.

We are grateful to James Giesecke who suggested that having different names will allow separate subtotals solutions to show the effects of the different homotopy variables.

1. You can now use the default statement

   `Equation (Default=ADD_HOMOTOPY=<default-homotopy-name>) ;`

   to set up the default name <default-homotopy-name> to be used for the homotopy levels variable to be added to any levels equations which follow. The Release 8.0 statement

   `Equation (Default=ADD_HOMOTOPY) ;`

Formula COMTOIND("c2") = "i7" ;
after the READ (BY_ELEMENTS) in the example, and this formula may alter the value of COMTOIND("c2").

[67] These sorts of Display statements were intended to work in Release 8.0 (as implied by the Release 8.0 documentation). But there were several bugs (for which we apologise) which prevented them working.

[68] This applies to versions of AnalyseGE built after September 2004. [See Help..About for the build date of your AnalyseGE.]

(which is still allowed without "=" after ADD_HOMOTOPY) is equivalent to

```
Equation (Default=ADD_HOMOTOPY=HOMOTOPY) ;
```

2. You can use the Equation qualifier

   ADD_HOMOTOPY=<homotopy-variable-name>

   to specify the name of the Levels Homotopy variable to be added to any levels Equation. The Release 8.0 qualifier ADD_HOMOTOPY (which is still allowed without a following "=") is equivalent to the qualifier ADD_HOMOTOPY=HOMOTOPY.

When TABLO comes to a levels equation,

- if there is an ADD_HOMOTOPY or ADD_HOMOTOPY=<name> qualifier in the statement the appropriate name is used for the levels homotopy variable added to the linearised equation.
- if a Default=Add_Homotopy statement is in force, and there is no ADD_HOMOTOPY or ADD_HOMOTOPY=<name> qualifier on the equation, the name specified in the Default statement is used for the levels homotopy variable added to the linearised equation.

The homotopy variable is always a levels variable. The **associated linear variable** always represents the change in that levels variable and the name of the associated linear variable always has **c_** added to the start of the levels name.

The default statement

```
Equation (Default=NOT_ADD_HOMOTOPY) ;
```

cancels any default name set up (as well as cancelling the adding of a homotopy variable to following equations).

You must add the relevant homotopy variables to the exogenous list in your Command file (you can use either the levels name or the associated linear name) and you must give them all a shock of size 1.

Example

Consider

```
Equation (Default=ADD_HOMOTOPY=homo1) ;
Equation (Levels) eq1  <etc> ;
Equation (Levels, Add_Homotopy=homo2) eq2 <etc> ;
Equation (Default=NOT_ADD_HOMOTOPY) ;
Equation (Levels, Add_Homotopy) eq3 <etc> ;
Equation (Levels) eq4 <etc> ;
```

The levels homotopy variables added are:

- homo1 to eq1,
- homo2 to eq2,
- HOMOTOPY to eq3.

No homotopy variable is added to eq4.

When you carry out a simulation, you should add homo1, homo2 and HOMOTOPY to the exogenous list and you should give each of these three variables a shock equal to 1.

8.7 Clarifications Relating to Release 8 Documentation

Below we clarify a few points that were omitted or were not made clear in the Release 8 documentation about TABLO Input files in GPD-2.

8.7.1 Set Statements

Set Equality

See section 3.1.2 of GPD-2. This relates to statements of the form

```
SET <set_name> = <set_name1> ;
```

This Set statement automatically generates the obvious two SUBSET statements, namely

```
Subset <set_name> Is Subset Of <set_name1> ;
Subset <set_name1> Is Subset Of <set_name> ;
```

In Release 8, both sets were required to be non-intertemporal sets. In Release 9, a Set Equality statement can be used to convert a non-intertemporal set to an intertemporal set, and vice versa – see section 3.3.

Data-Dependent Sets

See section 3.1.3 of GPD-2. This relates to statements of the form

```
Set <new_set> = (All, <index>, <old_set> : <condition> ) ;
```

- Here <old_set> must be a non-intertemporal set. [Intertemporal sets are not allowed.]
- This statement automatically generates the obvious Subset statement, namely

```
Subset <new_set> Is Subset Of <old_set> ;
```

Declaration of Intertemporal Sets

See case (5) in section 3.1 of GPD-2. Here either

```
MAXIMUM SIZE <integer>
```

or

```
SIZE <integer>
```

is allowed (though neither is required).

Examples (see section 7.2 of GPD-2)

```
Set (Intertemporal) fwdtime MAXIMUM SIZE 100
      ( p[0[ - p[NINTERVAL-1] ) ;
Set (Intertemporal) endtime SIZE 1 ( p[NINTERVAL] ) ;
```

Set Products

See section 3.1.4 of GPD-2. This relates to statements of the form

```
Set <new_setname> = <setname1> X <setname2> ;
```

Here both <setname1> and <setname2> must be non-intertemporal sets. [Intertemporal sets are not allowed.]

8.7.2 Formula Statements

See section 3.8 of GPD-2.

Labelling information (between # and #) is allowed. It must go after any qualifiers and before any quantifiers.

The qualifier BY_ELEMENTS is allowed if the left-hand side Coefficient is a Set Mapping – see section 3.13 of GPD-2 and section 8.4.1 above.

Writing Updated Values

This relates to Formula statements which begin

```
Formula (Initial, Write Updated Value To File <logical_filename>
    Header "<headername>" LongName "<words>")
```

- The logical file <logical_filename> can be either a text file or a Header Array file.
- If <logical_filename> is a Header Array file, the Header part is required and the LongName part is optional.
- If <logical_filename> is a text file, neither Header nor LongName is allowed.

8.7.3 Write Statements

See section 3.7 of GPD-2.

The qualifier BY_ELEMENTS is allowed if the Coefficient being written is a Set Mapping – see section 3.13 of GPD-2 and section 8.4.2 above. If that qualifier is present, the values are written as character strings, otherwise they are written as integers.

8.7.4 Equation Statements

See section 3.9 of GPD-2.

Note that the qualifiers ADD_HOMOTOPY and NOT_ADD_HOMOTOPY are only allowed in levels equations. They are not allowed in a linear equation.

See also section 8.6 above.

CHAPTER 9

9. Enhancements to Simulation Programs

Here we document various enhancements to the simulation programs (GEMSIM, TABLO-generated programs and SAGEM) for Release 9.0 and later. Included here are enhancements to the programs (including ViewSOL and GEMPIE) which show simulation results.

This is a long chapter but each of the sections is fairly short.

Other (major) enhancements to the simulation programs are described in chapters 2-5 and 7.

9.1 SSE Output

9.1.1 SSE Output from GEMSIM and TABLO-generated Programs

SSE Output is available from GEMSIM and TABLO-generated programs.

The FILE qualifier "**SSE**" which is an alternative to the qualifier "SPREADSHEET" – see section 3.5 of GPD-2 – is allowed in TABLO Input files for Release 9.0 and later.

Example.

```
File (New, Text, SSE)  Output1  # SSE output of coefficients # ;
```

If "SSE" is specified, output to the file is suitable for reading into a spreadsheet file. Matrices in the output have row and column elements labels.

The output is similar to SSE output from SLTOHT (see section 9.3 of GPD-4) and from SEEHAR (see the example in section 5.1.2 of GPD-4).

Example of SSE Output from TG-program or GEMSIM

RegResults1(REG:RegVar1)	u	y	EV
AUS	0.053	0.216	157.499
NAM	0.285	-0.628	16467.627
E_U	0.207	-0.446	12159.798
JPN	-0.051	-0.119	-1614.914
NIE	-0.897	-0.901	-4505.593
ASN	0.993	2.432	3540.311
.....			

9.1.2 SSE Output has Coefficient in Top LHS Corner

Tables of matrices produced with option SSE have the relevant part of the Coefficient or Variable shown in the top left-hand corner. For example, the table for qo(NSAV_COMM,REG) shown in section 9.3 of GPD-4 now has first row

qo(NSAV_COMM,REG), USA, EU, ROW

This applies to SSE output from SLTOHT, SEEHAR, GEMSIM and TABLO-generated programs.

9.2 Topics Relating to Simulation Accuracy (GEMSIM and TG-programs)

9.2.1 Smiling or Frowning Face Numbers are Reported

If you use WinGEM, RunGEM or RunGTAP, you know that after each simulation is completed, you see smiling or frowning faces which indicate how accurately the simulation was solved.

At the end of the simulation, and at the end of each subinterval if there is more that one, GEMSIM and TABLO-generated programs report (to the terminal and to the LOG file) the face number (between 1 and 10, where 10 is accurate and 1 is very inaccurate). Separate face numbers are shown for the variable and the data accuracy. If there is more than one subinterval, the face number for the overall variable accuracy is shown at the end of the Overall Accuracy Summary (see section 7.3.1 of GPD-3).

9.2.2 Fatal Error if Accuracy Too Low

If the accuracy is extremely low, it is highly unlikely that the simulation results are of any value. Accordingly, the program stops with a fatal error in order to draw your attention to this poor accuracy.

This can happen at the end of the whole simulation, or it can happen at the end of any subinterval which is solved inaccurately. However it does not happen at the end of a subinterval if you are using automatic accuracy since then you may redo the subinterval in question.

We believe it is your responsibility to decide whether or not a simulation has been solved sufficiently accurately. You should always look at the information provided by the software, especially the Accuracy Summaries (see sections 7.2 and 7.3.1 of GPD-3). We are reluctant to intervene and, as when equations are not satisfied very accurately (see section 12.6.1 of GPD-3), we end with a fatal error only in what we consider are extreme cases. At present, if any of the face numbers is 3 or less, we stop with a fatal error. Choosing 3 here does not mean that we think face numbers 4,5 and 6 are ok (since often they will not be) – rather we leave the decision to you in such cases.

9.3 Levels Results for Change Linear Variables

In Release 8.0, the last of the levels results shown (for example, by ViewSOL, RunGEM or GEMPIE) indicates the change in the levels value. When the linear variable in question is a Change variable, this just repeats the simulation result, so provides no additional information.

For example, consider a levels variable X declared in the TAB file via the statement

```
Variable (Levels, Change) X ;
```

Then the associated linear variable is c_X and this shows the change in the value of X over the simulation. Suppose that X changes from its pre-simulation value of 10 to a post-simulation levels values of 10.5. Then the levels results reported by ViewSOL will be

	(sim)	**pre-sim**	**post-sim**	**change**
c_X	0.5	10.0	10.5	0.5

The result in the last column just repeats the simulation result.

GEMPIE would show these results underneath each other as:

0.5 (the simulation result – the change)
10.0 (pre-sim levels value)
10.5 (post-sim levels value)
0.5 (the change)

9.3.1 GEMPIE and SLTOHT

For Release 9 and later, GEMPIE and SLTOHT have the option

PCL %-change for change var in levels results

If you select this option, for each linear variable which is a change variable (such as c_X in the example above), instead of showing the change as the last levels result, GEMPIE or SLTOHT will show the percentage-change as the last levels result. In other words, the fourth column shows the opposite (change/percent-change) of what the first column (the simulation result) shows.

For c_X in the example above, GEMPIE will show

0.5 (the simulation result – the change)
10.0 (pre-sim levels value)
10.5 (post-sim levels value)
5.0 (the percentage change)

If the pre-simulation levels value is zero, the percentage change cannot really be calculated but the program reports the percentage change as

- zero if the post-simulation value is also zero, or
- the very large number 10,000,000,000 if the post-simulation value is not zero.

Option PCL is selected by default in GEMPIE but not in SLTOHT.

This option PCL has no effect on linear variables which report percentage changes. For them the last of the levels results is always the change in the level of the variable.

9.3.2 ViewSOL

When ViewSOL built after December 2004 shows levels results, it behaves as if option PCL were selected. That is, the fourth column of levels results shows ordinary changes for variables which are percent-change in the TAB file and it shows percent changes for variables which are Change variables in the TAB file. In other words, the fourth column shows the opposite (change/percent-change) of what the first column (the simulation result) shows. This is true even if the Solution file you open was produced by Release 8.0 or earlier of GEMPACK. The header for this fourth column begins with **Ch/%Ch** to remind you.

9.3.3 RunGEM

When RunGEM built after December 2004 shows levels results, you can choose from the Options menu what the fourth column of levels results shows for change linear variables. This is true even if the Solution file you open was produced by Release 8.0 or earlier of GEMPACK. The relevant menu item under the Options menu is

Show %-change for Change Var in last Levels result

If this is selected, the heading for the last of the levels results will be something like **Chng/%-Ch** instead of the current **Chng**. The %-Ch part is to try to remind you that the results in this column will actually be percent changes if the variable in question is a Change variable.

9.4 Closure Specified Using Data Files

See section 5.2.5 of GPD-3. In Release 8.0, the data had to be read from a text file.

For Release 9.0 and later, the data can also be read from a Header Array file. To do that you include the word **header** and then specify the header between double quotes "" as in, for example,

```
Exogenous p_xfac = nonzero value on file sjclos1.har header "0002" ;
```

We are grateful to Glyn Wittwer for suggesting this enhancement.

Note that the software only checks that the total amount of data at the relevant position on the file agrees with the number of components of the variable in question. The actual dimensions of the data on the file need not match the dimensions of the variable. For example, suppose that the variable has two arguments of sizes 4 and 6 respectively (so that the variable has 24 components). There is no error if the data on the file is a 3x8 matrix, or a 2x4x2 matrix.[69]

[69] This is also how the checking is done in Release 8.0 (though that is not spelled out in the Release 8.0 version of GPD-3).

9.5 SLC, UDC and AVC Files

9.5.1 SLC Files May Contain Updated Values

SLC files (see section 8.4 of GPD-3) always contain the pre-simulation values of all Coefficients. The pre-simulation values of each Coefficient are shown at the header corresponding to the Coefficient number. For example, the pre-simulation values of the twentieth coefficient declared in the TAB file will be at header "0020".

If GEMSIM or a TABLO-generated program does any postsim passes (see chapter 2), the program also puts the updated (that is, post-simulation) values of all Coefficients onto the SLC file. The updated values of the twentieth coefficient declared in the TAB file will be at header "U020" on the SLC file. If there are more than 999 Coefficients, the first letter goes up through U,V and so on. For example, the updated values of coefficient number 2304 declared in the TAB file will be at header "W304" on the SLC file. The associated long name on the SLC file is "[Updated] CCC" where CCC is replaced by the name of the coefficient (for example, "[Updated] DVHOUS").

Example

> Use ViewHAR to look at the SLC file SJPSLB.SLC produced by the SJPSLB.CMF simulation you ran in section 2.8.1. For example, you will see the pre-simulation values of Coefficient XC at header 0007 and the post-simulation (or updated) values of Coefficient XC at header U007.

When a postsim pass is done, the SLC file contains all the information on the UDC file as well as the pre-simulation values of the coefficients. This makes the UDC file somewhat redundant (see section 2.9.3).

9.5.2 UDC and AVC Files Written on a Postsim Pass

UDC files contain the updated (that is, post-simulation) values of Coefficients while AVC files contain the average of the pre-simulation and post-simulation values of Coefficients (see section 8.5 of GPD-3).

In Release 8.0, the post-simulation values were calculated by carrying out an extra pass at the end of each separate multi-step calculation, and then by extrapolating from these values near the end of the run (see section 8.5.1 of GPD-3). In Release 9.0 and later, the post-simulation values and the average values are calculated during the first Postsim pass. If you request the UDC file to be created, the program always does at least one Postsim pass in order to calculate the post-simulation values of the coefficients.

9.5.3 SLC, UDC and AVC Files Contain Values of Postsim Coefficients

The values of any Postsim Coefficients (that is, Coefficients declared in the Postsim part of the TAB file – see section 2.5) are put onto all of the SLC, UDC and AVC files. For example, if the 65th coefficient declared in the TAB file is in a Postsim part of the TAB file, its values will be at header "0065" on the SLC, UDC and AVC files. The associated long name on the SLC/UDC/AVC file is "CCC [Postsim]" where CCC is replaced by the name of the coefficient.

AnalyseGE uses this information to show the values of postsim Coefficients (see section 2.9).

9.6 Checking for Duplicate Elements in a Set

We have made the checking for duplicate elements in a set more efficient. This more efficient checking is used when

- you read the elements of a set from a Header Array file as in, for example,
```
Set COM Read Elements from File Setinfo Header "COM" ;
```
- you specify the elements of a set via a list in the TAB file as in, for example,[70]
```
Set COM (food, agriculture, cars, manufactures, services) ;
```

9.7 When an Assertion Fails

If an Assertion fails, you will see, on the terminal and on the LOG file (if you produce a LOG file)

- the list of quantifier values for which the assertion fails.
- a closing statement indicating that the Assertion failed.

For Release 9 and later, the values of all Coefficients (or Variables) that occur in the Assertion are written to a special Header Array file so that you can look at the relevant values. We hope that this will make it easier for you to understand why your Assertion failed.

- If you are carrying out a simulation, the name of this Header Array file is
  <SL4-name>-ASSERT-FAIL.HAR
  where <SL4-name> is the name (excluding suffix) of the Solution file.
- If you are not carrying out a simulation, the name of this Header Array file is
  GP<xxx>-ASSERT-FAIL.HAR
  where <xxx> will be replaced by something like XX1 or X23. The LOG file will include the exact name of this Header Array file.

The different Coefficients and Variables will be written to different headers. The long name associated with each header will tell you which coefficient and which assertion the values relate to.

Example

Consider the following Assertion.

```
Assertion # Coeff1+Coeeff2 < 20 #
     (all,c,COM) coeff1(c) + coeff2(c) < 10 ;
```

Suppose that this fails when c="c2" and also when c="c3". Then, in the LOG file for this run, you will see something like the following.

```
  %% Assertion 'coeff1+coeff2 < 10' does not hold
   when quantifier is "c2".

  %% Assertion 'coeff1+coeff2 < 10' does not hold
   when quantifier is "c3".
 (Written real array, size 3, header '0003'.)
 (Written real array, size 3, header '0004'.)
  [Have written coeff/var values to headers "0003" to "0004"
   of file "gpxx1-assert-fail.har".]

  Assertion 'coeff1+coeff2 < 10' does not hold.
```

---

[70] You may notice a small change in reporting duplicate names in this case. In Release 9.0 and later, checking of duplicate names is done when the closing ")" is received at the end of the list and at most one duplicate pair is indicated. In Release 8.0 and earlier, each name was checked as it was read and all duplicate pairs were identified.
The increased efficiency in checking duplicate elements in both cases in the text comes from using a hashing function. We are grateful to Mark Horridge for writing the first version of this code.

The values of Coefficient COEFF1 are written to header "0003" and the long name at that header is "Values of coeff1 in assertion coeff1+coeff2 < 10". The last part of this is taken from the labelling information (between # and #) for the Assertion.

The values of Coefficient COEFF2 are written to header "0004".

9.8 Assertions, Writes on Preliminary Pass and Postsim Pass 2

9.8.1 Assertions and Writes to Terminal May Be Done on Preliminary Pass

GEMSIM and TABLO-generated programs always do a Preliminary pass in order to work out sets, subsets and set mappings (see section 13.2 of GPD-3 for details).

In Release 8.0, no Writes or Assertions were carried out on the Preliminary pass. This made it very difficult to identify any errors (such as arithmetic overflow) which may have occurred during the Preliminary pass. We are grateful to Robert McDougall for pointing out this problem.

For Release 9.0 or later,

- **writes to the terminal** (not to other places) are carried out during the Preliminary pass provided at least some values have been given to the Coefficient being written on the Preliminary pass.
- **assertions** are carried out during the Preliminary pass provided that at least some values have been given to **all** of the Coefficients involved in the Assertion on the Preliminary pass.

These Writes and Assertions will also be done again on the first actual pass.

Note that Coefficients are only given values on the Preliminary pass if their values are needed in order to work out the elements of a data-dependent set (see section 3.1.3 of GPD-2).

Example. Consider the following TAB file.

```
file input ;
set com read elements from file input header "COM" ;
coefficient (all,c,com) coef1(c) ;
coefficient (all,c,com) coef3(c) ;
read coef1 from file input header "c1" ;
read coef3 from file input header "c3" ;
! Next SHOULD be done on Preliminary pass !
assertion #check COEF1 values# (all,c,com) coef1(c) > 0 ;
write coef1 to terminal ;
! Next should NOT be done on Preliminary pass !
assertion #check COEF3 values# (all,c,com) coef3(c) > 0 ;
write coef3 to terminal ;
! Values of COEF1 will be worked out on Preliminary
     pass because of next statement !
set scom1 = (all,c,com: coef1(c) > 1) ;
write (all,c,scom1) coef3(c)  to terminal ;
```

The set SCOM1 is a data-dependent set. Hence the values of Coefficient COEFF1 will be worked out on the Preliminary pass. But the values of Coefficient COEFF3 will not be worked out on the Preliminary pass. Hence the Assertions and Writes to the terminal involving COEFF1 will be done on the Preliminary pass but not those involving COEFF3.

9.8.2 Assertions and Writes to Terminal May Be Done on Postsim Pass 2

When there are Postsim statements, sets and subsets declared in the Postsim part of the TAB file are worked out on the second postsim pass (see section 2.7).

- **Writes to the terminal** (not to other places) are carried out during Postsim pass 2 provided that at least some values have been given to the Coefficient begin written on Postsim pass 2. This applies to both normal and postsim Coefficients.

- **Assertions** are carried out during Postsim pass 2 provided that at least some values have been given to **all** of the Postsim Coefficients involved in the Assertion on Postsim pass 2. [Any normal Coefficients occurring always have values – their updated values – by this stage.]

These Writes and Assertions will also be done again on Postsim pass 3 (the final postsim pass).

Note that postsim Coefficients (those declared in a postsim part of the TAB file) are only given values on Postsim pass 2 if their values are needed in order to work out the elements of a data-dependent postsim set (see section 2.7). The values of ordinary Coefficients (those not declared in a postsim part of the TAB file) are always available during Postsim pass 2 since their values (updated values) are set up on Postsim pass 1.

9.9 Set and Element Names on Extrapolation Accuracy (XAC) Files

For Release 9.0 and later, we have tried to show set and element names in places where previously only component numbers were shown.[71]

In particular, we have done this on Extrapolation Accuracy files which now show

- the sets over which variables are declared,
- the labelling information (from the TAB file) for the variables,
- the element names for components of the variables associated with each line of results.

For example, the top of the results part of the Extrapolation Accuracy file SJLB.XAC from the standard SJLB.CMF simulation now looks as follows.

Excerpt from Extrapolation file SJLB.XAC

```
 -->  Variable p_Y  Total nominal household expenditure
p_Y          6.00000     5.94286      5.91412    5.88527  CX  4 L5

 -->  Variable p_PC(SECT)  Price of commodity i
p_PC(s1)     0.00000     0.00000      0.00000    0.00000  XMA 6 L6
p_PC(s2)    -1.00000    -0.973809    -0.961073  -0.948577 CX  2 L5

 -->  Variable p_XC(SECT,SECT)  Intermediate inputs of (etc)
p_XC(s1,s1)  6.00000     5.94286      5.91412    5.88527  CX  4 L5
p_XC(s2,s1)  7.00000     6.95000      6.92473    6.89929  CX  3 L5
p_XC(s1,s2)  6.00000     5.94286      5.91412    5.88527  CX  4 L5
p_XC(s2,s2)  7.00000     6.95000      6.92473    6.89929  CX  3 L5
```

If the variable name plus its components is too long (longer than about 25 characters) it goes on a line by itself and the results are on the following line.

[71] Another place where set and element names were introduced in Release 9.0 is discussed in connection with error messages in section 9.12.

9.10 Changes to How Extrapolation is Done

Suppose you are extrapolating from 3 separate multi-step calculations (for example, Gragg 2,4,6-step calculations).

For Release 8 and earlier the relevant extrapolation formula, as documented in Pearson (1991), was always used, even if the results appeared to be diverging or not converging well. This sometimes produced an extrapolated result that exaggerated the poor convergence, as the example below shows.

For Release 9 and later, we have changed this as follows.

- For bad convergence codes **OD!**, **MD!**, **OD?**, **MD?** and for not-so-bad codes **OC?** and **MC?**, the reported result is the result from the third multi-step calculation (the one with the most steps). The extrapolation formula is not used.
- For convergence codes which indicate that the three results are very close together – that is, for codes **MVC**, **OVC**, **MN0** and **ON0**, the reported result is the **average** of the three separate multi-step results. The extrapolation formula is not used.

The extrapolation formula is used in all other cases (as it was in Release 8).[72]

Example of Bad Convergence

The GC10E1.CMF simulation with GTAP61 extrapolates from Gragg 4,6,8-step calculations. Consider variable **CNTqgpdr(ASN)**.

The corresponding line in the Extrapolation Accuracy File produced with Release 8 is

```
    6        2.57380  0.661399  -0.143593   -1.28194   MC? 0
```

Clearly convergence is doubtful at best (as the code MC? indicates). The extrapolated result (this is what goes on the Solution file) is outside the range of the 3 separate results, and that only compounds the problem.

The corresponding line in the Extrapolation Accuracy File produced with Release 9 or later is

```
 CNTqpdr(ASN) 2.57380  0.661399  -0.143593   -0.143593  MC? 0
```

The extrapolated result is the same as the 8-step result. This does not remove the problem that the results are not converging. However it puts a less extreme result on the Solution file.

9.10.1 Shocked Variables

Consider the shocked variables. Irrespective of the number of steps taken, the accumulated result for a shocked component should be equal to the size of the shock apart from rounding errors.

If a large number of steps are taken, we have noticed that the accumulated results from the separate multi-step calculations are slightly different and also that the small differences between them are somewhat random. In such a case, the extrapolation formula may produce an extrapolated result that greatly exaggerates these random differences. This is one reason why we have changed the way extrapolation is done for codes MVC, OVC, MN0 and ON0 (see section 9.10 above).

[72] If DEBUG option number 194 is set, the extrapolation formula is always used (as it was in Release 8).

Example

Consider a shock of 10% and suppose that you extrapolate from 100,110,120-step Euler calculations. The accumulated results are

10.000002 [100 steps], 9.9999933 [110 steps] and 9.9999981 [120 steps].

These are all suitably close to 10. The convergence code is OVC (oscillating, very close together). So, as indicated in section 9.10 above, the Release 9 extrapolated result is the average of these, which will be 10.00000 (to 5 decimal places).

However, the extrapolation formula produces 10.000766, which is not helpful or satisfactory.

9.11 Accuracy Summary Information Now on Solution File

Accuracy Summary information (for variables and updated data – see section 7.2 of GPD-3) is now put onto Solution files produced by Release 9.0 or later versions of GEMSIM and TABLO-generated programs. This makes Accuracy Summary information available to the Windows programs including ViewSOL and AnalyseGE which can now show it.

If your simulation has only one subinterval,

- the Variable Accuracy Summary information put on the Solution file is as shown in section 7.2.1 of GPD-3,
- the Updated Data Accuracy Summary put on the Solution file is the overall accuracy summary for all updated data files (if two or more are updated) or else is the accuracy summary for the only updated file (if only one is updated). See section 7.2.2 of GPD-3 for more information about this.

If your simulation uses more than one subinterval,

- the Variable Accuracy Summary Information put on the Solution file is the Overall Accuracy Summary – see section 7.3.1 of GPD-3.
- there is no Updated Data Accuracy Summary put on the Solution file since, at present, the software does not report an overall accuracy for the updated data when two or more subintervals are used.

9.12 Swap Statements on Command Files

Names of levels variables are allowed in swap statements (see section 5.2.3 of GPD-3). [As suggested in section 5.10 of GPD-3, we intended to allow this in Release 8.0, but levels names were not allowed in swap statements due to an oversight in that release.]

If you include a swap statement which is not valid, error reports are now much clearer and more specific than they were in Release 8.0 or earlier. Error reports include set and element names in the components of the variables in question.

9.13 Command File Statements for Newton's Method

For information about Newton's method, see section 7.5 of GPD-3.

For Release 9.0 and later, there are several Command file statements to make running a simulation using Newton's method easier. The original method for using Newton's method (as described in section 7.5 of GPD-3) still applies but the Command file method is easier to use.

The difficulty with the original method was that during the simulation you were prompted for details about the Newton correction at each step in the multi-step simulation. In order to use Newton's method, the program (GEMSIM or the TABLO-generated program) had to be run interactively (or from a Stored-input file) so that you could respond to prompts at each step.

In the Command file method, all the information for Newton's method can be supplied using the following Command file statements:

```
method = newton ;
newton shock = YES | no ;      !optional, default = yes
newton steps-per-euler = <d1> ; !optional, default <d1> = 2
newton extra-steps-at-end = <d2> ; !optional, default <d2> = 4
```

Examples

1. The simplest way is to just change the method to newton, and specify the number of (Euler) steps. (For example, see SJLVLBNW.CMF in section 9.13.2 below.)

   ```
   method = newton ;
   steps = 4 ;
   ```

   The simulation uses the default values so the actual steps are:
   1 Euler step (with Newton correction added) then 2 Newton steps
   1 Euler step (with Newton correction added) then 2 Newton steps
   1 Euler step (with Newton correction added) then 2 Newton steps
   1 Euler step (with Newton correction added) then 6 Newton steps

2. If you want to specify more closely the number of Newton steps you could use:

   ```
   method = newton ;
   steps = 1 ;
   newton shock = no ;  ! Do not add Newton correction to Euler steps
   newton steps-per-euler = 0 ;
   newton extra-steps-at-end = 3 ;
   ```

   The simulation will do 1 ordinary Euler step (with no Newton correction included in the simulation shocks), then 4 Newton correction steps after it. (For example, see SJLVLBNW2.CMF in section 9.13.3 below.)

Differences between the Command file and original Newton simulation methods

The Comomand file and original methods are not compatible with each other. This is because the original method asks questions at every step whereas the Command file method does not ask questions since all the Newton information is in the Command file.

In both methods

1. you need a levels model.
2. you must run TABLO and select option NWT at the first option screen (see section 2.1 of GPD-2).
3. you then run GEMSIM or the TABLO-generated program (after compiling and linking).
4. you must make $del_Newton an exogenous variable, for example, include the following statement in your Command file: "exogenous $del_Newton ;"
5. you do not need to add a shock statement for the variable $del_Newton in your Command file.

Original method (see section 7.5 of GPD-3)

Run the program interactively (or from a Stored-input file).

Either select NWT at the first option screen of GEMSIM or the TG program, or use the Command file statement "NWT=yes ;"

You can use a Command file for the remainder of the simulation information.

You must use "method = euler ;"

Answer questions at each Euler step about Newton's corrections.

Command file method

Run the program using a Command file including the Newton Command file statements.

Use "method = newton ;" (Don't use the NWT option).

Command file statements for Newton's method are documented in section 9.13.1 below. We describe simple examples in sections 9.13.2 and 9.13.3 below.

9.13.1 Details of the Newton's Method Command file Statements

`method = newton ;`

This is a new method which is an alternative to the usual methods Euler, midpoint, Gragg and Johansen. This statement sets the method to Newton's method.

When the method is set to Newton using "method = newton ;" no questions are asked during the separate steps of the simulation. The program either uses values from the other Newton Command file statements listed below, or if they are omitted, it uses the default values.

With "method = newton ;", the option NWT or Command file statement "NWT=yes;" is not allowed.

The simulation does several Euler steps as specified by the usual "steps = ...;" statement. You can only use a single multi-step solution, for example "steps = 5 ;" You can not use 2 or 3 multi-steps followed by extrapolation (as in "steps = 1 2 4 ; !wrong") because extrapolation does not work when you are using Newton's method. Within the program, the method used is really Euler and the "steps=.." command says how many Euler steps are to be used. The extra Newton steps are specified below or from the defaults.

`newton shock = YES | no ;` `! optional, default is "yes"`

If you have "newton shock = yes ;" or this statement is omitted, the program will add a Newton shock (see section 7.5 of GPD-3) to the variable $del_Newton when carrying out each of the Euler steps.

If you have "newton shock = no;", the usual shocks for Euler's method will be applied at each Euler step.

This statement can only be used in combination with "method=newton;".

`newton steps-per-euler = <d1> ;` `! optional, default <d1> = 2`

The integer <d1> gives the number of Newton steps carried out after each Euler step. If this statement is not present after "method = newton ;" then the default value of <d1> = 2 is used.

This statement can only be used in combination with "method=newton;".

`newton extra-steps-at-end = <d2> ;` `! optional, default <d2> = 4`

The integer <d2> gives the number of extra Newton steps carried out after the final Euler step, so that the number of Newton steps after the final Euler step will be <d1> + <d2>. If this statement is not present after "method = newton ;" then the default value of <d2> = 4 is used.

This statement can only be used in combination with "method=newton;".

9.13.2 Example: SJLVLBNW.CMF

In the GEMPACK examples, there are two Command files using Newton's method: SJLVLBNW.CMF and SJLVLBNW2.CMF.

The levels model used is SJLV using the TABLO-input file SJLV.TAB.

First run TABLO taking inputs from the Stored-input file **SJLVNWGS.STI** (which selects option NWT at the start and which produces output files for GEMSIM called **SJLV-NWT.GSS** and **SJLV-NWT.GST**).

Then run GEMSIM taking inputs from the Command file **SJLVLBNW.CMF**.[73]

Notice that the Newton correction variable $del_Newton is declared as exogenous in these Command files.

In Command file SJLVLBNW.CMF, the default values of various Newton's command file statements are used. As in example 1 in section 9.13, the simulation carries out:

1 Euler step (with Newton correction added) then 2 Newton steps
1 Euler step (with Newton correction added) then 2 Newton steps
1 Euler step (with Newton correction added) then 2 Newton steps
1 Euler step (with Newton correction added) then 6 Newton steps

Part of Command file SJLVLBNW.CMF

```
auxiliary files = sjlv-nwt ;

! Rel 9 command file statements for Newton's method

method = newton ;
steps = 4 ;

! Closure
exogenous p_xfac ;
! for Newton's method $del_Newton must be exogenous
exogenous $del_Newton ;
rest endogenous ;
```

Have a look at the LOG file SJLVLBNW.LOG produced to see the different Euler and Newton steps.

[73] You will also need the data file SJ.DAT.

9.13.3 Example: SJLVLBNW2.CMF

In the Command file SJLVLBNW2.CMF, the simulation will carry out a 4-step Euler calculation (with no Newton's corrections). Then following the 4-step Euler, Newton's method is used to increase the accuracy of the simulation by applying the 3 extra Newton steps at the end.

Part of Command file SJLVLBNW2.CMF

```
!  Auxiliary files for model SJLV-NWT (from SJLV.TAB)
!  In TABLO, choose option NWT
auxiliary files = sjlv-nwt ;

! Rel 9 command file statements for Newton's method

method = newton ;
steps = 4 ;
! carry out  4 step Euler with no Newton corrections, then
! 3 extra steps at the end where the only shock is Newton correction
newton shock = no ;             ! the default is yes
newton steps-per-euler = 0 ;    ! default is 2
newton extra-steps-at-end = 3 ; ! default is 4

report newton errors = all ;    ! default is initial


! Closure
exogenous p_xfac ;
! for Newton's method $del_Newton must be exogenous
exogenous $del_Newton ;
rest endogenous ;
```

First run GEMSIM taking inputs from the Command file **SJLVLBNW2.CMF.**

Then look at the LOG file SJLVLBNW2.LOG to see where the Euler and Newton steps are done.

The statement

```
report newton errors = all ;
```

(see section 9.14) in the Command file asks the program to report information about the Newton errors after each Euler and Newton step.

Look at the log file SJLVLBNW2.LOG to see what is happening to the errors as the simulation progresses. (Search for "del_Newton".) You can see the Newton error terms are initially zero, then they increase during the Euler steps. After the fourth Euler step and before the first Newton step at the end, the

Sum of absolute values of the del_Newton terms is 0.479.

During the 3 subsequent Newton steps, the errors decrease to very small values:

Sum of absolute values of the del_Newton terms is 2.5629 E-04.
Sum of absolute values of the del_Newton terms is 6.5565 E-05.
Sum of absolute values of the del_Newton terms is 1.5497 E-04.

9.14 Reporting Newton Error Terms in Models with Levels Equations

If your model has all (or many) levels equations, you can report errors in these levels equations during and after a simulation. This provides an independent check as to the accuracy (or otherwise) of your simulation.

In order to be able to report the errors, you must select option NWT (see section 2.1 of GPD-2) at the first option screen when you run TABLO. Then TABLO adds the **Newton correction variable** $del_Newton to your model and adds so-called **Newton error terms** (or Newton correction terms) to the linearised version of each levels equation.

The size of the Newton error term tells you how well each levels equation is satisfied.

- If the Newton error term is zero or small, the equation is satisfied well.
- If the Newton error term is not small, the equation is not satisfied well.

As indicated in section 7.5.3 of GPD-3, you can check the accuracy of the initial database by reporting the sizes of the Newton error terms as calculated from the initial data.

With the Command file statements described below, you can now report the sizes of all or selected Newton error terms as calculated from the final updated data or as calculated from any of the updated data sets produced during a multi-step calculation.

You do not need to be using Newton's method in order to be able to report these errors. You can also report them if you are using the traditional GEMPACK solution methods Euler, midpoint or Gragg (as the example in section 9.14.1 below makes clear).

Note that the Newton errors can only be calculated and reported for equations which are written as Levels equations in your TABLO Input file. They cannot be reported for linearised equations in the TABLO Input file.

Note also that you must include the Newton correction variable $del_Newton as an exogenous variable in your Command file (perhaps by including the statement "exogenous $del_newton ;").

We first describe an example in section 9.14.1 below. Then, in section 9.14.2 below, we fully document the Command file statements you can use to request reports about Newton error terms.

9.14.1 Example: Reporting Newton Errors with SJLVLBER.CMF

This is the usual Stylized Johansen simulation where the supply of labor is increased by 10% while the supply of capital is held fixed. It is run with the levels version SJLV.TAB (see section 3.5.1 of GPD-1) of the Stylized Johansen model instead of the mixed version SJ.TAB (see section 3.3 of GPD-1).

First run TABLO taking inputs from the Stored-input file **sjlvnwgs.sti** (which selects option NWT at the start and which produces output files for GEMSIM called **sjlv-nwt.gss** and **sjlv-nwt.gst**).

Then run GEMSIM taking inputs from the Command file **sjlvlber.cmf**. Notice that the Newton correction variable $del_Newton is declared as exogenous in this Command file. This simulation uses Euler 1,2,4 steps.

When this simulation has run, look at the LOG file **SJLVLBER.LOG** in TABmate.

First search for "**based on initial data**". You will see the lines

```
 Newton error terms (based on initial data).
 Sum of absolute values of the del_Newton terms is 0.0000000.
  The norm of this del_Newton terms vector is 0.0000000.
```

This tells you that all the levels equations in the model are satisfied exactly (up to the precision used by the program) since the sum of the absolute values of the del_Newton terms (these are the Newton error terms) is zero.

Then search for "**based on updated data**". You will see lines like the following.

```
  Final Newton error terms (based on updated data).
  Newton error = -1.19209290E-05 in equation Comin("s2","s2").
  Newton error = -8.34465027E-05 in equation Facin("labor","s1").
  Newton error = -4.76837158E-05 in equation Facin("labor","s2").
  Newton error = -2.38418579E-05 in equation House("s2").
<etc>
  Sum of absolute values of the del_Newton terms is 6.91890717E-04.
   The norm of this del_Newton terms vector is 2.88324867E-04.
  Largest $del_Newton term is in equation Com_clear("s2").
  The $del_newton term in this is -2.44140625E-04.
```

These are the values of the nonzero Newton error terms when calculated using the updated data. They show how accurately the levels equations are satisfied using the solution values reported by GEMSIM. Happily all of the errors are small so you can see that this is quite an accurate solution. The "Sum of absolute values of the del_Newton terms" value is the sum of all Newton errors. And the largest error is also reported (and the equation which produces it).

The Command file SJLVLBER.CMF contains the statement

```
Report newton errors = all ;
```

which asks GEMSIM to report the sizes of the Newton error terms at all steps.

Go back to the top of the LOG file SJLVLBER.LOG and search for "**Newton error** =". You will see something like the following.

```
  --->  Beginning pass number 2 of 2-pass calculation.

  Newton error terms (as at the start of this simulation step).
  Newton error = -1.69873238E-02 in equation Comin("s2","s1").
  Newton error = -1.69873238E-02 in equation Comin("s2","s2").
  Newton error = -9.71794128E-02 in equation Facin("labor","s1").
<etc>
  Sum of absolute values of the del_Newton terms is 0.50708652.
   The norm of this del_Newton terms vector is 0.19212985.
  Largest $del_Newton term is in equation Facin("labor","s1").
  The $del_newton term in this is -9.71794128E-02.
```

These report the sizes of the Newton error terms after the first step of the 2-step Euler calculation. So these report the sizes after half of the full shock has been given in a single step. Notice that the errors are not inconsiderable. Notice also the value of the sum of all errors and of the largest one.

If you keep searching for "**Newton error** =" you will see similar reports which relate to the solution after 1, 2 or 3 of the 4 Euler steps carried out as the third multi-step calculation. Note that the sum of all errors keeps increasing. [It is about 0.13 after the first step, 0.25 after the second step and about 0.37 after the third step. Although the sum of all errors is not shown after the fourth step you can see that it would be about 0.5.]

Fortunately the extrapolation works well and (as you saw above when you searched for "based on updated data") the errors in the levels equations are very small after extrapolation.

As you will see in section 9.14.2 below, you can control how often these Newton error reports are given.

9.14.2 Command File Statements for Reporting Newton Error Terms

To generate reports about the Newton correction terms in the levels equations of your model, you must select the initial option NWT (see section 2.1 of GPD-2) when running TABLO.

The following Command file statement allows you to see reports about the sizes of the Newton error terms at various stages during your simulation, including (most importantly) at the start and at the end. The sizes of these Newton error terms tell you how well the levels equations in your model are satisfied. In particular, if you report the values at the end of the simulation, you can tell whether or not you have solved the model sufficiently accurately.

Note that the Newton errors can only be calculated and reported for equations which are written as Levels equations in your TABLO Input file. They cannot be reported for linearised equations in the TABLO Input file.

```
Report newton errors = INITIAL | final | subinterval | all | none ;
                       !  optional, default is "initial"
```

You can only choose one of the above alternatives (initial, final, subinterval, all or none).

```
Report newton errors = initial ;
```

prints out the Newton error terms based on the initial data.

```
Report newton errors = final ;
```

prints out the Newton error terms after the simulation has finished (based on the updated data and on the solution values in the Solution file).

```
Report newton errors = subinterval ;
```

prints out the Newton error terms after each subinterval and at the end.

```
Report newton errors = all ;
```

prints out the Newton error terms at the start (initial), at the start of each simulation step, at the start of each subinterval during the run, and after the simulation has finished.

By default, if there are Newton correction terms available (by running TABLO with the NWT option), the initial errors (those based on the initial data) will be reported in all the above cases. To turn off reporting Newton errors entirely, use the statement

```
Report newton errors = none ;
```

The following Command file statement allows you to include in these reports just the Newton errors greater than a specified number:

```
Report newton min_error = <r1> ; ! optional, default <r1> = 0
```

where r1 is a real number. For example if you include a statement in your Command file

```
Report newton min_error = 10 ;
```

the program will only print out information about equations for which the Newton correction term is greater than 10. It will also print out the name of the equation and components where the Newton correction term occurred.

The LOG file discussed in section 9.14.1 above should make the effect of these different statements clear.

Other Examples

In sections 9.13.2 and 9.13.3 above you carried out simulations using Newton's method. The Command files included the statement

```
Report newton errors = all ;
```

so the sizes of the error terms were reported after every step (Euler or Newton). Look again at the LOG files **SJLVLBNW.LOG** and **SJLVLBNW2.LOG** produced to see how the errors change after the different steps.

9.15 Suppressing Arithmetic Errors in GEMSIM and TABLO-generated Programs

Normally if GEMSIM or a TABLO-generated program encounters an arithmetic problem (overflow etc), the program stops with a fatal error – see section 15.6 of GPD-3.

For Release 9.0 or later, if you include the statement

```
arithmetic problems = warn ;  ! alternative is "= fatal ;" which is the default
```

in your Command file for GEMSIM or a TABLO-generated program, the program will just give a warning if an arithmetic error occurs and continue running. It will not set a fatal error at the end of the run, but will report how many arithmetic problems have occurred during the run.

Of course, if the program encounters arithmetic errors, at least some of the results are likely to be non-sensible. It is your responsibility to check this. And some other programs (for example, ViewHAR and ViewSOL) may crash when you try to view the results since some numbers may not be legal values.

We are grateful to Ernesto Valenzuela for passing on a simulation which showed that the option to suppress arithmetic problems might be useful in some cases.

9.16 Rate% Shock Statement

This is a shock statement intended for shocks to powers of taxes or subsidies.[74]

The syntax of the statement is

```
shock <v1 [component_list]> = rate% <value> ;
```

This means to change the *ad valorem* rate by <value> percent. Note that <value> is not a change in the *ad valorem* rate, but a percentage change.[75] You should read the examples below carefully to be sure that you understand the effect of this rate % shock statement.

Here the variable v1 must be a percent-change variable (not a change variable). Pre-simulation levels values for this variable must be available, either

- because v1 is a levels variable, or
- because v1 is a linear variable associated with a levels variable, or
- because v1 is a linear variable whose pre-simulation levels value has been specified via an ORIG_LEVEL= qualifier (see sections 3.4 and 4.5.5 of GPD-2) when the variable was declared.

Note that the word "rate%" can only be used in "shock" statements. It cannot be used in "ashock" or in ‘tshock" statements (see section 5.5.4 of GPD-3). Nor can it be used in "final_level", "change" or "percent_change" statements – see sections 5.6 and 5.7 of GPD-3.

You can use this statement if you wish to increase or decrease some *ad valorem* tax rates by some uniform percent. The examples below will make this clear.

For example, suppose that you want to model a 50% trade liberalization. Then you could use "rate% –50" in the relevant shock statement.

Recall that the connection between the *ad valorem* rate (V) and the power (P) is (in the levels):

$$P = 1 + V / 100$$

For example, if the *ad valorem* rate V is 25% then the power P is 1.25.

---

[74] This type of shock statement was added in Release 8.0-001 (October 2003). It is not documented in the Release 8.0 documentation, so is included here.

[75] The effect of this type of shock statement is the same as the effect of the option "%change rate" offered on the Shocks page of RunGTAP for variables such as `tms` and `to` in the GTAP model.

There are some limitations as to when you can use the rate% shock statement – see section 9.16.1 below.

Example 1. In GTAP.TAB (see, for example, GTAP61.TAB supplied with the GEMPACK examples), the linear variable **to(i,r)** represents the percentage change in the power of the tax or subsidy on production of commodity i in region r. It could be declared via the statement[76]

```
Variable (ORIG_LEVEL=TO_L)  (all,i,TRAD_COMM)(all,r,REG)  to(i,r)
 # power of tax or subsidy on output of commodity i in region r # ;
```

Here TO_L(i,r) is a Coefficient whose values are equal to VOA(i,r)/VOM(i,r).

Suppose that there are just three commodities, food, manufacture, services and suppose that the pre-simulation values of TO_L(i,"EU") in the European Union EU are

1.3 (food) 1.1 (manufacture) 0.9 (services)

Consider the statement

```
shock to(TRAD_COMM,"EU") = rate% 20 ;
```

This tells the software to apply shocks to the variables `to(TRAD_COMM,"EU")` so as to increase the associated *ad valorem* rates by 20 percent (not 20 percentage points).

1. For food, the pre-simulation power is 1.3 so the pre-simulation rate is 30%. Increasing this rate of 30 by 20 percent will make it 36%. Hence the post-simulation power must be 1.36. This means that the shock applied to the variable to(food,EU) will be
   100*(.06/1.3) = 4.615 percent.
2. For manufacture, the pre-simulation power is 1.1 so the pre-simulation rate is 10%. Increasing this rate of 10 by 20 percent will make it 12%. Hence the post-simulation power must be 1.12. This means that the shock applied to the variable to(manufacture,EU) will be
   100*(.02/1.1) = 1.818 percent.
3. For services, the pre-simulation power is 0.9 so the pre-simulation rate is –10%. Increasing this rate of –10 by 20 percent will make it –12%. Hence the post-simulation power must be 0.88. This means that the shock applied to the variable to(services,EU) will be
   100*(–.02/0.9) = –2.222 percent.

Now consider the statement (this time with a negative shock)

```
shock to(TRAD_COMM,"EU") = rate% -20 ;
```

This tells the software to apply shocks to the variable to(TRAD_COMM,"EU") so as to decrease the associated *ad valorem* rates by 20 percent.

1. For food, the pre-simulation power is 1.3 so the pre-simulation rate is 30%. Decreasing this rate of 30 by 20 percent will make it 24%. Hence the post-simulation power must be 1.24. This means that the shock applied to the variable to(food,EU) will be
   100*(–.06/1.3) = –4.615 percent.
2. For manufacture, the pre-simulation power is 1.1 so the pre-simulation rate is 10%. Decreasing this rate of 10 by 20 percent will make it 8%. Hence the post-simulation power must be 1.08. This means that the shock applied to the variable to(manufacture,EU) will be
   100*(–.02/1.1) = –1.818 percent.
3. For services, the pre-simulation power is 0.9 so the pre-simulation rate is –10%. Decreasing this rate of –10 by 20 percent will make it –8%. Hence the post-simulation power must be 0.92. This means that the shock applied to the variable to(services,EU) will be
   100*(.02/0.9) = 2.222 percent.

Example 2. The Miniature Orani model in the GEMPACK examples MO.TAB contains a levels variable T(i) which is the power of the import tariff on commodity i.

Here we discuss the well-known simulation described in section 8.4 of Dixon *et al* (1982). The purpose of that simulation is to increase the *ad valorem* tariff rate on commodity "C2" by 25 percent.

---

[76] The ORIG_LEVEL part is not included in the standard versions of GTAP.TAB.

Command file **MOTAR.CMF** is the traditional way of giving that shock. To prepare that Command file, the appropriate shock to give to the power of the tariff was calculated (using values in the database). The shock statement in MOTAR.CMF is

```
shock p_T = 0 7.353 ;
```

This gives a 7.353 percent increase to the power of the tariff on commodity "C2".

Command file **MOTAR2.CMF** uses the shock rate% statement being documented here to give the same shock. The shock statement in MOTAR2.CMF is

```
shock p_T("C2") = rate% 25 ;
```

To see that these two are the same, we suggest that you run the simulations in these two Command files MOTAR.CMF and MOTAR2.CMF, and then look at the results in ViewSOL. Under the ViewSOL menu File | Options, make sure the box marked "Show levels results (if present)" is checked.

Look at the MOTAR2.SL4 results for the for the variable p_T.

p_T	motar2	Pre motar2	Post motar2	Chng motar2
c1	0	1.1111	1.1111	0
c2	**7.3529**	**1.4167**	**1.5208**	**0.1042**

The results show that the power of the tariff T for commodity "C2" is 1.4167 pre-simulation. Hence the pre-simulation *ad valorem* tariff rate for commodity "C2" is 41.67%. Post-simulation, after increasing the *ad valorem* tariff rate by 25%, the new *ad valorem* rate is 52.08% and the new power of the tariff is 1.5208. Hence the power T of the tariff on commodity "C2" increases from 1.4167 to 1.5208. This is a percentage increase in the power T of 7.3529%, as you can see if you use a calculator.

Note that this value 7.3529 is approximately the same as the shock to p_T("C2") (7.353) in the original Command file MOTAR.CMF.

The rate% way is a more direct way of applying this shock. No prior calculation of the shock is required (as it was for MOTAR.CMF). And the **rate% 25** shock would work with any base data, not just the version supplied in MO.DAT.

9.16.1 Fine Print about Rate% Shock Statements

Consider the general rate% shock statement

```
shock <v1 [component_list]> = rate% <value> ;
```

GEMPACK knows whether the levels values associated with the variable v1 are available. It also knows whether the linear variable associated with v1 is a percentage change variable. If either of these is not true, GEMPACK will not allow the statement.

But GEMPACK does not know whether or not the variable v1 is the percent change in the power of some tax or subsidy. It is **your responsibility** to use rate% only in a shock statement when the associated levels quantity is the power of some tax or subsidy. You know the sort of formula GEMPACK will use to calculate the shock when rate% is in the statement and you must ensure that this is appropriate for the variable in your rate% shock statement.

The power of a tax or subsidy must be positive. [A value of zero would correspond to an *ad valorem* rate of –100%, while a negative value would correspond to a negative *ad valorem* rate of more than 100%.] Hence GEMPACK will report an error if

- the relevant levels variable (the one associated with linear variable v1) has pre-simulation value zero or negative.
- the post-simulation value for the relevant levels variable would be zero or negative after the shock is applied. [For example, suppose that the pre-simulation power is 0.4 (the *ad valorem* rate is then –60%) and that the statement is "shock v1 = rate% 100 ;". Then the rate –60 should be doubled according to the shock statement which would make the post-simulation *ad valorem* rate –120%, which is not allowed since it would make the post-simulation power negative.]

9.17 Shock Statements using a Slice from a Header Array file

What is a **slice** of an array?

An array is defined over several sets. A slice of the array is just part of the array corresponding to one element in one of the sets instead of the whole set.

> **Example**. Consider a three dimensional array ARRAY defined over SET1xSET2xSET3. Then one element can be chosen from one of the sets, for example, element "ssss11" from set SET1.
>
> Then the **slice** of ARRAY(SET1,SET2,SET3) corresponding to element "ssss11" in SET1 is the two dimensional array ARRAY("ssss11",SET2,SET3).

Prior to Release 9, as documented in section 5.5.2 of GPD-3, the dimensions of the array in a shock statement had to match the dimensions of the variable being shocked. This applies to all statements like the three below.

```
shock <variable> = file <shock-filename> header "ABCD" ;
shock <variable> <components> =
                      select from file <filename> Header "XIFC" ;
shock p_XINTFAC("capital",IND) =
                      select from file SHOCK.HAR header "XINT";
```

In the last of the examples above, `SHOCK.HAR` is a Header Array file containing a Header `"XINT"`. Since the variable p_XINTFAC in Stylized Johansen is defined over the sets FACxIND, the array on the file at this header must also be of size FACxIND.

For Release 9.0 or later, this statement is extended so that a **slice** of a Header array can be used instead of a complete Header.

The syntax is as follows:

```
shock <variable> = file <shock-filename> header "<header-name>"
                                              slice "<element-name>" ;

shock <variable> <components> = select from file <shock-filename>
                  header "<header-name>" slice "<element-name>" ;
```

Example

```
shock p_XINTFAC("capital",IND) = select from file SHOCK.HAR
                                    header "XINT"  slice  "Y2005" ;
```

> If the slice part of the statement was omitted, the Header "XINT" should contain an array of size FACxIND. When using the slice of Header XINT, the array can be of dimension FACxINDxYEAR where YEAR is a set containing the element "Y2005". The new statement above then uses the slice FACxINDx"Y2005" of the Header array "XINT" instead of the whole array.

We call the element after the "slice" part of the statement the **slice element**. [For example, the slice element is Y2005 in the last example above.]

The slice syntax applies to all similar shock-like statements, that is to statements with keywords "ashock", "tshock", "final_level", "change" or "percent_change" – see chapter 5 of GPD-3.

Example – Several Related Simulations

Suppose that you are carrying out several simulations with different sized shocks to the same variables in each case. Then you might use the set of simulations as the extra dimension and put the different shocks to one variable on the same header. For example, you may have the set

SIMS = (LowGrowth, MedGrowth, FastGrowth)

representing the simulations. If you wish to shock an industry variable AIND(IND) of technological changes (one for each industry) in each simulation, you could build an array of size IND x SIMS at header "AIND". This one array will contain the different shocks for each industry and each simulation. In the Command file for the low growth simulation you will put the statement

```
shock aind = file <filename> header "AIND" slice "LowGrowth" ;
```

while you will put

```
shock aind = file <filename> header "AIND" slice "FastGrowth" ;
```

in the Command file for the fast growth simulation.

This will reduce the number of separate shocks files you need to maintain. Also you can see at once the different shocks for each simulation (rather than having to look in 3 separate files).

9.17.1 Use with RunDynam, RunMONASH and related Windows programs

Slices can be useful if you are doing a set of simulations covering a sequence of years, for example 2005, 2006 to 2020. You could have a set such as (Y2005, Y2006,….Y2020) to indicate the different years.

The same Header can be used to hold the shocks for all simulations and the slice element can be used to select one at a time. For example, the header on the file could be of size FACxINDxYEAR where YEAR is the set that contains elements (Y2005, Y2006,….Y2020).

In the first simulation (the one for year 2005), the shock statement would be

```
shock p_XINTFAC = file SHOCK.HAR header "XINT" slice "Y2005" ;
```

In the second simulation (the one for year 2006), the shock statement would be

```
shock p_XINTFAC = file SHOCK.HAR header "XINT" slice "Y2006" ;
```

and so on.

In BSH and PSH files prepared for use with RunDynam etc, the word **<year>** can be used to instruct the windows program to add the slice part when preparing Command files for each year. For example,

- the statement in the BSH or PSH file

  ```
  shock <year> p_XINTFAC = file shock.har header "XINT" ;
  ```

  will tell RunDynam to put in the Command file for year 2005

  ```
  shock p_XINTFAC = file shock.har header "XINT" slice "Y2005";
  ```

  and to put in the Command file for year 2006

  ```
  shock p_XINTFAC = file shock.har header "XINT" slice "Y2006";
  ```

- the statement in the BSH or PSH file

  ```
  shock <year=ABC> p_XINTFAC = file shock.har header "XINT" ;
  ```

  will tell RunDynam to put in the Command file for year 2005

  ```
  shock p_XINTFAC = file shock.har header "XINT" slice "ABC2005";
  ```

  and to put in the Command file for year 2006

  ```
  shock p_XINTFAC = file shock.har header "XINT" slice "ABC2006";
  ```

See the RunDynam Help file for more details.

We are grateful to James Giesecke for suggesting that we allow shocks to be read from a slice in an array.

9.17.2 Syntax Rules for Slice Shock Statements

- In the Header, there are the same sets as usual but there is one extra dimension (the **slice dimension**).
- The extra dimension can occur anywhere.
  Eg FACxINDxYEAR or FACxYEARxIND or YEARxFACxIND
- The Header must be of type 'RE' (that is, it must contain real data with set and element labelling information). Each set at the header must have known elements (see section 4.6.1 of GPD-2) and the slice element must be an element of the slice dimension.
- The **slice element** (the one after the word "slice") must be an element of exactly one of the sets.

9.18 SSL and Subtotals

If you have

- subtotal statements in your Command file and
- an SSL (save Separate multi-step SoLutions – see section 7.7.1 of GPD-3 – these solutions are saved as subtotal solutions) statement in your Command file,

the genuine subtotals (corresponding to the subtotal statements in your Command file) are the first subtotals put on the Solution file and the SSL solution(s) are the last subtotal(s) put on the Solution file.[77]

For example, if your Command file contains the statements

```
method = euler ;
steps = 2 4 6 ;
ssl = last ;
subtotal p_XFAC("labor") = labor effects ;
```

then there will be two subtotals results on the Solution file. The first is the "labor effects" subtotal and the second is the SSL solution which is the 6-step Euler solution. [The subtotal description for this will be "Results from the 6-step solution".]

9.19 Solution, Equations and Environment Files are Header Array Files

Solution, Equations and Environment files produced by Release 9.0 or later GEMPACK programs (including GEMSIM, TABLO-generated programs and SAGEM) are Header Array files.

We have made these files Header Array files mainly to make it easier for us to maintain and develop the code. A secondary benefit is that Windows programs such as ViewSOL, RunGEM and AnalyseGE can read Solution files directly – they do not have to use SLTOHT. [See "What's New" in the current Help files for more details.] This makes reading Solution files in ViewSOL much faster.

Although ViewHAR (or SEEHAR) can read these files, you still need to use ViewSOL or GEMPIE to look at simulation results on a Solution file in a meaningful way since the arrays on the file are not written to be looked at directly.

[77] There was a bug in Release 7 and Release 8 of GEMPACK which meant that you could never see both the genuine subtotals results and the SSL results. Then two headers 'STLS' were written on the file, one for the genuine subtotals results and one for the SSL results. It was a matter of chance which of these GEMPIE or SLTOHT found.

9.20 Several Subintervals – Clarification

You can only specify more than one subinterval if you are extrapolating from 2 or 3 separate multi-step calculations. We apologise that this was not made clear in the Release 8.0 (and earlier) documentation – see section 7.3 of GPD-3.

In Release 8.0, if you specified several subintervals but were only doing a single multi-step calculation, the software (GEMSIM or TABLO-generated program) said it was ignoring the subintervals statement and one subinterval was used. For Release 9.0 and later, we have changed this and now a fatal error occurs in that case.

CHAPTER 10

10. Enhancements to Utility Programs

Here we document enhancements to the various utility programs – see GPD-4 – for Release 9.0. Some of these apply to all GEMPACK programs for Release 9.0 and later – see sections 10.1 and 10.2.

10.1 New Command Line Options for Some Programs

See section 5.5 of GPD-1 for details about the various command line options in Release 8.

When you use the command line options described below, if you want to specify a file name which includes spaces, put the file name in quotes as in

```
tablo "c:\my dir\sj"
```

10.1.1 Options –log, –lon and –los

In Release 8, these options were only allowed when you also had either –cmf or –sti on the command line (see section 5.5 of GPD-1). For Release 9.0 and later, you can specify a log file using –log, –lon or –los any time, even if you are running the program interactively. For example:

```
tablo  -log sjtab.log
```

10.1.2 TAB file and WFP/PGS on Command Line with TABLO

If you do not want to do any condensation actions not included in the TAB file, and you do not want to specify any options other than PGS or WFP at the Code stage, you can include the name of the TABLO Input file on the command line. TABLO then runs in batch mode and does not expect any other input. If there are condensation actions in the TAB file, running with **–pgs** or **–wfp** will do those condensation actions (but give no opportunity for other condensation actions) and then go to Code stage.

- You specify the name of the TABLO Input file on the command line – don't include the suffix .TAB.
- You can specify PGS by putting –**pgs** on the command line.
- You can specify WFP by putting –**wfp** on the command line.

Examples

1. To run TABLO to process SJ.TAB

   ```
   tablo  sj
   ```

   This will produce either a TABLO-generated program or output for GEMSIM, depending on which is the default action for the TABLO.EXE which is running. [The default is TABLO-generated program if TABLO.EXE is from a Source-code Version of GEMPACK or is GEMSIM if TABLO.EXE is from an Executable-Image Version of GEMPACK.]

2. To run TABLO to process SJ.TAB and produce output for GEMSIM

   ```
   tablo  sj  -pgs
   ```

3. To run TABLO to process SJ.TAB and produce output for GEMSIM and (see section 10.1.1) send all output to file SJGS.LOG

```
tablo  sj  –pgs  –log sjgs.log
```

4. To run TABLO to process SJ.TAB and produce the TABLO-generated program SJ.FOR

```
tablo  –wfp  sj
or
tablo sj  -wfp      the order is not important
```

If you specify the TABLO Input file on the command line, it is the same as running TABLO interactively and hitting carriage-return for every response (once you have supplied the TAB file name). However you still retain the WFP/PGS choice.

Note that including **–pgs** or **–wfp** on the command line only affects the way TABLO runs if you also specify the name of the TAB file on the command line.

10.1.3 SLTOHT – Solution File on Command Line

With the program SLTOHT, you can use

```
sltoht  <infile>
```

Here **<infile>.sl4** is the name of the input Solution file. This produces Header Array output (the default for SLTOHT – see chapter 8 and section 8.6 of GPD-4). The name of the output file is **<infile>.sol**. This output file contains values from the cumulative solution. No other user input is required. Note that you must not include the suffix .SL4 in <infile>.

Example

```
sltoht  sjlb
```

processes the cumulative solution on SJLB.SL4 and produces Header Array output SOL file SJLB.SOL.

10.1.4 CMPHAR and Files on the Command Line

With the program CMPHAR, you can use

```
cmphar  <infile1>  <infile2>  <outfile>
```

Here <infile1> and <infile2> are the names of the two Header Array files you wish to compare. Outfile is the name of the output print file which contains information about the differences between the two Header Array files. You can also omit <outfile> as in

```
cmphar  <infile1>  <infile2>
```

in which case the output file has the same name as infile1 with the last dot in the name replaced by a hyphen "-" and with the suffix .CMH. For example, if you type in

```
cmphar  sj1.dat  sj2.dat
```

then the output file is called **SJ1-DAT.CMH**.

As usual, if a file name contains spaces, enclose it in quotes. For example,

```
cmphar  "my sj\sj1.dat"  sj2.dat
```

10.2 Deleting Temporary Lahey/Fujitsu Copies of Binary Files

As explained in section 15.2 of GPD-4, GEMPACK programs sometimes make temporary copies of binary files. This is because of the differences between Lahey and Fujitsu versions of files.

These temporary copies are usually put into the directory pointed to by the Environment variable TEMP on your computer (see section 15.2.1 of GPD-4).

When a GEMPACK program from Release 9.0 or later completes its run, it deletes any temporary copies created during the run.[78] [With Release 8.0, these temporary copies were left in the TEMP directory. Perhaps you should look in that directory. If you find any files of the form G*.TMG, you can safely delete them since they are temporary copies made by earlier runs of Release 8 or earlier GEMPACK programs.]

10.3 Suppressing Arithmetic Errors in ACCUM, DEVIA and CMBHAR

Normally, when the programs ACCUM, DEVIA or CMBHAR encounter an arithmetic problem (such as overflow – see section 15.6 of GPD-3), they stop with a fatal error.

The most likely cause of overflow is having a percentage-change variable which should have been declared as a CHANGE variable in the TABLO Input file.

If you select option

ANF Arithmetic problems not fatal

in any of these programs, the program attempts to continue even if arithmetic problems occur. Instead you will see a warning. If an arithmetic problem occurs, or if a result goes outside what we consider to be a "safe" range, a somewhat arbitrary value will be given in the output file. In these cases, the programs attempt to signal the problem to you as follows.

- ACCUM adds "Arithmetic Problem" in the last column of the output spreadsheet file in every row for which an arithmetic problem was encountered.
- DEVIA does the same as ACCUM except that there is no space between "Arithmetic" and "Problem".
- CMBHAR adds the words "ARITH PROB" at the start of the long names of all arrays or variables for which an arithmetic problem was encountered.

Versions of RunDynam, RunMONASH etc after July 2004 have the option "Arithmetic problems not fatal for spreadsheet jobs" under the Run Preferences menu. When this option is selected, the windows programs runs ACCUM, DEVIA and CMBHAR with option ANF during the spreadsheet jobs.

10.4 SLTOHT

SLTOHT can access a single subtotal solution – see section 6.1.1.

[78] TABLO-generated programs and GEMSIM tell you at the end if they have deleted a temporary copy of one of the input data files. Otherwise, GEMPACK programs do not tell you (in the LOG file) when they have deleted a temporary copy.

10.5 CMPHAR Reports Difference Metric Values

CMPHAR is used to compare the values on two Header Array files – see section 4.2 of GPD-4.

If you think that the values should be nearly the same, it was difficult with the Release 8 version of CMPHAR to decide about this just by looking at the summary statistics shown at the end of the CMPHAR LOG or output file. The problem was that,

- for large values, small differences are not important and so the difference ratio is a better measure. For example if the two values are 1000000 and 1000002, the difference is 2 which does not seem small but the difference ratio is about 0.000002 which does seem small.
- for small values, the difference ratio can be large even when the difference is small. For example, if the two values are 0.000001 and 0.000003 then the difference is only 0.000002 but the difference ratio is 0.666667 which does not seem small. For small numbers the difference is the better measure.

So, for small values, the difference is the better measure while for large numbers the difference ratio is the better measure.

This is why CMPHAR now reports what we call the **difference metrics**. The difference metric is a measure of the difference between two different numbers say V1 and V2.

- If either V1 or V2 is less than 1 in absolute value, the difference metric is just the difference. [Notice that this includes the case where V1 or V2 is zero and the other is nonzero.]
- Otherwise (that is, if both V1 and V2 are both at least 1 in absolute value), the difference metric is the difference ratio which, as before, is equal to ABS(V1-V2)/MIN[ABS(V1), ABS(V2)].

Then, if you expect the values on the two files to be close, you will expect the total of all the difference metrics to be small and similarly for the average of the difference metrics. Both of these numbers are shown at the end of a CMPHAR run. You can use just these to decide if the values on the two files are essentially the same.

Examples

If V1=2 and V2=3, the difference metric is the difference ratio, namely (3-2)/2=0.5.

If V1=0.9 and V2=3, the difference metric is the absolute difference, namely 3-0.9=2.1.

If V1=0.6 and V2=0, the difference metric is the absolute difference, namely 0.6-0=0.6.

10.6 SEEHAR Option FNP

In SEEHAR, option FNP (Fewer new pages) – see section 4.1.1 of GPD-4 – is now set true by default. [You can turn it off in the usual way by responding "-fnp" when you see the options menu.]

CHAPTER 11

11. GEMPACK Windows Programs

When you install Release 9 of GEMPACK you will also need to install the associated Windows programs (WinGEM, ViewHAR, ViewSOL, TABmate etc, etc) from your Release 9 GEMPACK CD. You will need to use Release 9 versions of all of these programs as soon as you start using Release 9 of GEMPACK (Source-code or Executable-Image Versions). For example, Release 9 Solution files cannot be read by the Release 8 version of ViewSOL.

11.1 MIN Files Not Needed by RunGEM for Release 9 Programs

AXT (Auxiliary Table files) and GST (GEMSIM Table files) for Release 9 (or later) TABLO-generated programs or GEMSIM are Header Array files.

RunGEM can read these files directly. Since these files contain all the information on the Model Information file (.MIN file), RunGEM does not require you to supply the MIN file if your model has been processed by TABLO from Release 9 or later.

See "What's New" in the current RunGEM Help file for more details.

11.2 ViewSOL and Release 9 Solution files

Release 9 Solution files are Header Array files – see section 9.19.

ViewSOL can read a Release 9 Solution file directly. ViewSOL does not need to run SLTOHT first. Hence opening a Release 9 Solution files in ViewSOL is much faster than opening a Release 8 or earlier Solution file.

Of course the Release 9 ViewSOL can still read Solution files produced by Release 8 or earlier of GEMPACK. In this case, ViewSOL still needs to run SLTOHT in the background to convert the Solution file to a SOL file.

11.3 AnalyseGE

See chapter 6 and section 2.9 for details of important new features.

11.4 Enhancements to Other Windows Programs

Other enhancements to the Windows programs are described in sections 9.3 and 9.17.1 of this document.

There have been many enhancements which are not described in this document to the GEMPACK windows programs since the Release 8 versions of these programs. See "What's New" in the Help file for the different programs for details.

11.5 RunMONASH, RunDynam and RunGTAP

If you use RunMONASH, RunDynam or RunGTAP, you will need to make some other simple changes when you install Release 9 of GEMPACK. See chapter 12 for details.

CHAPTER 12

12. Release 9 of GEMPACK with RunMONASH, RunGTAP

If you use the Windows programs RunMONASH, RunDynam or RunGTAP, you will need to make a few simple changes in order for these programs to be able to work with TABLO-generated EXEs (or GEMSIM) made with Release 9 of GEMPACK (Source-code or Executable-Image Versions).

See section 12.1 for RunMONASH, RunDynam, RunGDYN and related programs.

See section 12.2 for RunGTAP.

12.1 RunMONASH, RunDynam etc

This section refers to RunMONASH and other similar Windows programs such as RunDynam, RunGDYN, RunQGEM, RunGTEM etc.

You will need to put Release 9 versions of various GEMPACK programs, including
SLTOHTA.EXE, SEENVA.EXE, ACCUMA.EXE,
DEVIAA.EXE, CMBHARA.EXE, TEXTBIA.EXE
into the directory in which your RunMONASH or RunDynam etc is installed.

For **RunMONASH**, if your Release 9 GEMPACK is installed in C:\GP9 and your RunMonash is installed in C:\RUNMON, copy the files as shown below, replacing the existing earlier versions.

```
copy  c:\gp9\accum.exe  c:\runmon\accuma.exe
copy  c:\gp9\devia.exe  c:\runmon\deviaa.exe
copy  c:\gp9\seenv.exe  c:\runmon\seenva.exe
copy  c:\gp9\sltoht.exe  c:\runmon\sltohta.exe
copy  c:\gp9\textbi.exe  c:\runmon\textbia.exe
copy  c:\gp9\cmbhar.exe  c:\runmon\cmbhara.exe
```

(If your Release 9 GEMPACK and RunMONASH are not installed in these directories, replace with the correct directories on your computer.)

For **RunDynam**, if your Release 9 GEMPACK is installed in C:\GP9 and your RunDynam is installed in C:\RUNDYNAM (replace with the correct directories for your computer),

```
copy  c:\gp9\accum.exe  c:\rundynam\accuma.exe
copy  c:\gp9\devia.exe  c:\rundynam\deviaa.exe
copy  c:\gp9\seenv.exe  c:\rundynam\seenva.exe
copy  c:\gp9\sltoht.exe  c:\rundynam\sltohta.exe
copy  c:\gp9\textbi.exe  c:\rundynam\textbia.exe
copy  c:\gp9\cmbhar.exe  c:\rundynam\cmbhara.exe
```

The Release 9 GEMPACK programs SLTOHTA.EXE, SEENVA.EXE etc are backwards compatible so RunMONASH and RunDynam etc using these GEMPACK programs should also work with

1. old TABLO-generated EXEs from Release 8 and 7 (as stored in your ZIP archives) and
2. new TABLO-generated EXEs made with the new Release 9 GEMPACK.

12.2 RunGTAP

You will need to put Release 9 versions of the programs SLTOHTA.EXE, SEENVA.EXE, RWSLA.EXE, REPSOL.EXE and SAGEM.EXE into the directory in which your RunGTAP is installed. The first two are necessary for simulation while the second two are necessary if you want to carry out Systematic Sensitivity Analysis.

- For the first two (SLTOHTA.EXE and SEENVA.EXE) you can copy the files SLTOHT.EXE and SEENV.EXE from the directory in which you have installed Release 9 of GEMPACK (adding "A" to the end of their names when you put them in the RunGTAP directory). [For example, copy SLTOHT.EXE from the directory in which Release 9 of GEMPACK is installed to the directory in which RunGTAP is installed, and call the copy SLTOHTA.EXE, overwriting the existing SLTOHTA.EXE there.]
- For RWSLA.EXE, if you have a Source-code version of Release 9 of GEMPACK, go to a DOS box in the directory in which Release 9 of GEMPACK is installed and type in the command

  mkmain rwsl

  This will make RWSL.EXE. Then copy RWSL.EXE to the directory in which RunGTAP is installed and give it the name RWSLA.EXE there.
- For RWSLA.EXE, if you have an Executable-Image version of Release 9 of GEMPACK, look on the GEMPACK web site. A suitable version of RWSLA.EXE will be available there.
- For SAGEM.EXE, you can copy the file SAGEM.EXE from the directory in which you have installed Release 9 of GEMPACK.
- For REPSOL.EXE, look on the GEMPACK web site. A suitable version of REPSOL.EXE will be available there.

With these changes, we believe that moderately old versions of RunGTAP (for example, Version 3.20 of RunGTAP dated June 2003) will handle Release 9 TG-programs perfectly.

CHAPTER 13

13. New Release 9 TABLO Statements and Qualifiers

FILE qualifier "SSE" is an alternative to qualifier "SPREADSHEET" – see section 9.1.

```
File (New, Text, SSE)  <logical-filename> #<labelling-info># ;
```

READ qualifier "IfHeaderExists" is allowed – see section 8.1.

```
Read (IfHeaderExists) <coeff> from File <filename> Header "...." ;
```

WRITE or DISPLAY qualifier "Postsim" is allowed – see section 2.5.

```
Write (PostSim) <coefficient/variable> to file ..... ;
Display (PostSim) <coefficient/variable>  ;
```

"**Postsim (Begin) ;**" and "**Postsim (End) ;**" statements mark the beginning and end of Postsim sections – see section 2.5.

The syntax and semantic rules for the Postsim sections of a TAB file are set out in section 2.5.

Default statement for lower/upper bound on coefficient (as set out in section 8.2):

```
COEFFICIENT(DEFAULT= LOWER_BOUND GE <real-number> ) ;
COEFFICIENT(DEFAULT= LOWER_BOUND GT <real-number> ) ;
COEFFICIENT(DEFAULT= LOWER_BOUND OFF ) ;
COEFFICIENT(DEFAULT= UPPER_BOUND LE <real-number> ) ;
COEFFICIENT(DEFAULT= UPPER_BOUND LT <real-number> ) ;
COEFFICIENT(DEFAULT= UPPER_BOUND OFF ) ;
```

Qualifier BY-ELEMENTS is no longer required in a Formula for a Set Mapping if there is an element or an index-expression on the RHS – see section 8.4.1.

Writing part of a Set Mapping BY_ELEMENTS is allowed – see section 8.4.2.

Display statements (simple or complicated) for an Integer Coefficient of dimension 3 or higher. [See section 8.5.]

Simple Write statements (not allowed to involve quantifiers, elements or subsets) for an Integer Coefficient of dimension 3 or higher. [See section 8.5.]

Statements converting an intertemporal set to a non-intertemporal set, and vice versa (see section 3.3.1):

```
Set (Intertemporal) <new-set> = <old-non-intertemporal-set> ;
Set (Non_Intertemporal) <new-set> = <old-intertemporal-set> ;
```

Ranked sets. [See section 3.2]

```
SET <new-set> [#<labelling-info>#] =
          <old-set> Ranked UP|DOWN BY <coeff/var-name> ;
```

CHAPTER 14

14. New Release 9 Command File Statements

14.1 GEMSIM, TABLO-generated Programs and SAGEM

start with MMNZ2 = <integer value> ; ! see sections 5.1 and 4.2.6

LU decompose transpose = yes|NO ; ! see section 5.3

Markowitz pivots in MA48 = yes|NO ; ! see section 5.4

MA48 compress = yes|NO|few ; ! see section 5.2.1

MA48 increase_MMNZ = slow|medium|FAST|veryfast ; ! see section 5.2.2

shock <variable> = file <shock-filename> header "<header-name>"
slice "<element-name>" ;

shock <variable> <components> = select from file <shock-filename>
header "<header-name>" **slice "<element-name>" ;**

! see section 9.17

14.2 Just GEMSIM and TABLO-generated Programs

share_memory yes|NO ; ! see section 4.2

ma48 use_original = yes|NO ; ! see section 5.1.1

Various Command file statements relating to Newton's method – see section 9.13.1
and reporting Newton errors - see section 9.14

method = newton ;
newton shock = YES | no ; !optional, default = yes
newton steps-per-euler = <d1> ; !optional, default <d1> = 2
newton extra-steps-at-end = <d2> ; !optional, default <d2> = 4
Report newton errors = INITIAL | final | subinterval | all | none ; ! default is "initial"
Report newton min_error = <r1> ; ! optional, default <r1> = 0

complementarity approx_as_subtot = first|last|yes|no ; ! see section 7.1

complementarity subtotals = approximate|accurate; ! see section 7.1

arithmetic problems = warn|FATAL ; ! see section 9.15

Postsim extra statements including
XPostsim (Begin) ;
XPostsim (End) ; ! see section 2.6

Postsim = YES|no ; ! see section 2.7

15. REFERENCES

Dixon P.B., B.R. Parmenter, J. Sutton and D.P. Vincent (1982), *ORANI: A Multisectoral Model of the Australian Economy*, North-Holland, Amsterdam.

Duff, I.S. and J.K. Reid (1993), 'MA48, a Fortran Code for Direct Solution of Sparse Unsymmetric Linear Systems of Equations', Rutherford Appleton Laboratory Report RAL-93-072, pp.62.

Harrison, W.J., J.M. Horridge and K.R. Pearson (2000), 'Decomposing Simulation Results with Respect to Exogenous Shocks', *Computational Economics*, vol.15, pp.227-249. [A preliminary version was *Centre of Policy Studies and Impact Project Preliminary Working Paper* No. IP-73, May 1999.]

Harrison, W.J., J.M. Horridge and K.R. Pearson (2005), 'Using GEMPACK Subroutines in Your Fortran Programs'; to be presented at the GTAP conference in Lubeck, Germany in June 2005. This is available at *http://www.monash.edu.au/policy/gpusesub.htm*

Pearson K.R. (1991), 'Solving Nonlinear Economic Models Accurately via a Linear Representation', *Impact Preliminary Working Paper* No. IP-55, Melbourne (July), pp.39.

16. 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.

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

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.

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

Harrison, W.J. and K.R. Pearson (2005), *Release 9 of GEMPACK: New Features and Changes from Release 8.0*, GEMPACK Document No. 5 **[GPD-5]** Monash University, Clayton, First edition, April 2005.

Harrison, W.J. and K.R. Pearson (2005), *Installing and Using the Source-Code Version of GEMPACK on Windows PCs with Lahey Fortran*, GEMPACK Document No. 6 **[GPD-6],** Monash University, Clayton, 12th edition, April 2005.

Harrison, W.J. and K.R. Pearson (2005), *Installing and Using the Executable-Image Version of GEMPACK on Windows PCs*, GEMPACK Document No. 7 **[GPD-7],** Monash University, Clayton, 9th edition, April 2005.

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.

.

17. INDEX

$

$del_Comp
  For complementarities, 7-62
$del_Newton, 9-92, 9-94
$POS
  Use in ranking sets, 3-32

%

%change rate in RunGTAP, 9-97

A

ACCUM
  Documentation. *See* chapter 10 of GPD-4
Accuracy
  Faces (smiling or frowning), 9-80
Accuracy Summary, 9-80
  Available via AnalyseGE, 9-89
  Available via ViewSOL, 9-89
  Faces (smiling or frowning), 9-80
  On Solution file, 9-89
Accuracy too low
  Fatal error, 9-80
Accurate run
  Subtotals, 7-58
ADD_HOMOTOPY
  Default statement, 8-74
  Qualifier, 8-74
Address
  Contact, 1-1
Allowed file names. *See* chapter 5 of GPD-1
AnalyseGE, 6-55. *See* chapter 2 of GPD-4
  Decompose selected expression, 6-55
  Subtotals, 6-55
AnalyseGE example
  Data-manipulation TAB file. *See* section 6.8.2.1 of GPD-3
  Simulation. *See* section 6.1 of GPD-8
Approximate run
  Subtotals, 7-58
Arithmetic errors
  GEMSIM and TG-programs, 9-97
Arithmetic problems
  ACCUM, DEVIA, CMBHAR, 10-107
  GEMSIM and TG-programs, 9-97
Assertion
  Seeing values when fails, 9-84
Assertions
  On Postsim pass 2, 9-85
  On preliminary pass, 9-85
AVC file
  Written on Postsim Pass 1, 9-83
AXT file
  Now Header Array file, 11-109

B

Brockmeier, Martina, 1-4, 7-57
BY_ELEMENTS qualifier
  In Formulas, 8-73
  Writing Set Mappings, 8-73

C

Centre of Policy Studies. *See* chapter 1 of GPD-1
  Address, 1-1
Change levels results
  May be percent changes, 9-81
Choosing variables interactively. *See* chapter 17 of GPD-4
Closure
  Data on Header Array file, 9-82
  Specifying. *See* chapter 5 of GPD-3
CMBHAR
  Documentation. *See* chapter 4 of GPD-4
CMF file. *See* Command file
CMPHAR
  Checking if two files are the same, 10-108
  Command line, 10-106
  Difference metrics, 10-108
  Documentation. *See* chapter 4 of GPD-4
Coefficient
  Integer, dimension 3 - seeing values, 8-74
  Normal. *See* Normal Coefficient
  Ordinary. *See* Ordinary Coefficient
  Postsim. *See* Postsim Coefficient
  Specifying allowed range of values, 8-72
Coefficient qualifier Non_Parameter
  Ignored in postsim part, 2-23
Coefficient qualifier Parameter
  Ignored in postsim part, 2-23
Command file
  Documentation. *See* GPD-3
  Newton's method, 9-90
  On command line. *See* chapter 5 of GPD-1
  Statements. *See* GPD-3
  Syntax. *See* GPD-3
Command file statements
  Summary. *See* chapter 18 of GPD-3
Command line
  TABLO, SLTOHT, CMPHAR, 10-106
Command line options
  -log, -lon and -los, 10-105
Command prompt. *See* section 2.3 of GPD-1
Comments
  In Command files. *See* section 2.7 of GPD-3
  In GEMPACK text data files. *See* section 6.1 of GPD-4
  In TABLO Input files. *See* section 4.1 of GPD-2
Compile and link. *See* section 1.2 of GPD-2
  On Windows PCs. *See* section 6.3 of GPD-6
Complementarities. *See* chapter 16 of GPD-3
Complementarity
  Subtotals, 7-58
Complementarity overshooting
  Subtotal, 7-62
Component numbers. *See* section 5.3 of GPD-3
Compressions in MA48A, 5-49
CONLF. *See* chapter 15 of GPD-4
Contact
  GEMPACK, 1-1
ConvHAR. *See* chapter 15 of GPD-4

D

Data
  Transferring to/from GAMS. *See* chapter 16 of GPD-4
Decomposing results
  Meaning, 7-57
Default statements
  Not allowed in postsim part, 2-23
Deleting temporary copies of data files, 10-107
Demonstration Version. *See* chapter 1 of GPD-1
DEVIA
  Documentation. *See* chapter 10 of GPD-4
Difference metric
  CMPHAR, 10-108
Dixon, Peter, 4-40, 4-44
Duff, Iain, 5-51, 5-51

E

Environment files
  Now Header Array files, 9-102
Equal sets. *See* Set equality
Equations files
  Now Header Array files, 9-102
Error handling. *See* chapter 5 of GPD-1
Examples
  Hands-on. *See* GPD-8
Executable image
  Of a TABLO-generated program. *See* section 1.2 of GPD-2
Executable-image GEMPACK
  Installing on Windows PCs. *See* GPD-7
Executable-image Version. *See* chapter 1 of GPD-1
  Limited. *See* chapter 1 of GPD-1
  Unlimited. *See* chapter 1 of GPD-1
Extra statements, 2-24
Extrapolation Accuracy file. *See* section 7.2 of GPD-3
Extrapolation Accuracy File, 9-87
Extrapolation Accuracy Summary. *See* section 7.2 of GPD-3
Extrapolation formula, 9-88
  Not used in some cases, 9-88

F

Faces (smiling or frowning), 9-80
FAQs, 1-3. *See* chapter 1 of GPD-1
Ferreira Filho, Joaquim, 1-4
File names
  Allowed. *See* chapter 5 of GPD-1
  Containing spaces, 10-105
File suffixes. *See* chapter 5 of GPD-1
File types. *See* chapter 5 of GPD-1
Formula qualifier Always
  Ignored in postsim part, 2-23
Formula qualifier Initial
  Ignored in postsim part, 2-23
Francois, Joe, 1-4
Frequently Asked Questions, 1-3. *See* chapter 1 of GPD-1
Fujitsu files, 10-107
  Deleting temporary copies, 10-107
Fujitsu Header Array files. *See* chapter 15 of GPD-4
Full path names. *See* chapter 5 of GPD-1

G

GAMS data files. *See* chapter 16 of GPD-4
GEMPACK
  Demonstration Version. *See* chapter 1 of GPD-1
  Executable-image Version. *See* chapter 1 of GPD-1
  FAQs, 1-3. *See* chapter 1 of GPD-1
  Frequently Asked Questions, 1-3. *See* chapter 1 of GPD-1
  Hands-on introduction. *See* GPD-8
  Source-code Version. *See* chapter 1 of GPD-1
  Versions, 1-1
  World Wide Web, 1-3
GEMPACK Examples
  Associated files. *See* chapter 1 of GPD-8
GEMPACK licence, 1-1
GEMPACK licence file. *See* chapter 1 of GPD-1
GEMPACK programs, 1-2
  Common features. *See* chapter 5 of GPD-1
  List. *See* chapter 1 of GPD-1
GEMPACK text data files
  Documentation. *See* chapter 6 of GPD-4
GEMPACK Versions. *See* chapter 1 of GPD-1
GEMPACK web site. *See* chapter 1 of GPD-1
GEMPACK-L
  Mailing List, 1-3
GEMPACK-L Mailing List. *See* chapter 1 of GPD-1
GEMPIE
  Documentation. *See* chapter 7 of GPD-4
GEMSIM
  Documentation. *See* GPD-3
Giesecke, James, 8-74, 9-101
GST file
  Now Header Array file, 11-109
GTEM model, 5-51

H

Hands-on examples. *See* GPD-8
Hanslow, Kevin, 1-4
HAR file. *See* Header Array file
Header Array files
  Introduction. *See* section 3.1 of GPD-4
Hertel, Tom, 1-4
Homogeneity
  Nominal. *See* chapter 13 of GPD-4
  Real. *See* chapter 13 of GPD-4
Homogeneity simulations. *See* chapter 13 of GPD-4
Homotopy variables
  Names for, 8-74
Horridge, Mark, 3-34, 9-84

I

IfHeaderExists
  Read, 8-71
Impact Project, 1-1. *See* chapter 1 of GPD-1
Increasing MMNZ
  Inside MA48 routines, 5-50
INF file. *See* Information file
Installing Executable-image GEMPACK
  On Windows PCs. *See* GPD-7
Installing Source-code GEMPACK
  On Windows PCs. *See* GPD-6
Integer Coefficient
  Dimension 3 - seeing values, 8-74

Integer Coefficient dimension 3
Display, 8-74
Seeing values, 8-74
Write, 8-74
Intertemporal models. *See* chapter 7 of GPD-2
Intertemporal set
Converting to non-intertemporal, 3-35

J

Jacobsen, Lars-Bo, 1-4
Jakeman, Guy, 5-51

K

Ko, Jong-Hwan, 1-4

L

Lahey files, 10-107
Deleting temporary copies, 10-107
Lahey Header Array files. *See* chapter 15 of GPD-4
Levels equations
How accurately satisfied, 9-94, 9-96
Levels results
Change variables, 9-81
LHS Matrix
LU decompose transpose, 5-51
Licences for GEMPACK, 1-1
Limited Executable-image Version. *See* chapter 1 of GPD-1
Line length
In Command files. *See* section 2.7 of GPD-3
In GEMPACK text data files. *See* section 6.2 of GPD-4
In TABLO Input files. *See* section 4.1 of GPD-2
LOG files
On command line. *See* chapter 5 of GPD-1
Lorenz curve, 3-36
Losers, 3-31
LOWER_BOUND
In Default statement in TAB files, 8-72
LTG
For compiling and linking. *See* section 1.2 of GPD-2
LU decomposition
Of transpose of LHS Matrix, 5-51
Times, 5-51

M

MA48
Compressions, 5-49
Original, 5-48
Reallocate without starting LU again, 5-48
Speed of increasing MMNZ, 5-50
MA48AG, MA48BG
GEMPACK versions of MA48A,MA48B, 5-49
Macintosh PCs
Installing GEMPACK, 1-4
Mailing List
GEMPACK-L, 1-3
Markowitz Pivots, 5-52
McDougall, Robert, 8-74, 9-85
Memory
Reallocating, 5-48
Report, 4-39
Share, 4-40
Memory management. *See* chapter 5 of GPD-1
Method
Newton, 9-91
MIN file
May not be needed by RunGEM, 11-109
MKHAR
Documentation. *See* chapter 11 of GPD-4
MKSOL
Documentation. *See* chapter 11 of GPD-4
MMNZ
Increasing, 5-50
MMNZ, MMNZ1 and MMNZ2, 5-47
Model examples, 1-2
Model Information File. *See* MIN file
MODHAR
Documentation. *See* chapter 3 of GPD-4
MOIQ3.CMF
And related examples, 7-62

N

Newton
Minimum error, 9-96
Shock, 9-90
Steps at end, 9-90
Steps per Euler, 9-90
Newton correction terms, 9-94
Newton correction variable, 9-94
Newton error terms, 9-94
Newton's method
Command file, 9-90
Nominal homogeneity. *See* chapter 13 of GPD-4
Non-intertemporal set
Converting to intertemporal, 3-35
Normal Coefficient, 2-19, 2-20
Normal part
Of TAB file, 2-19, 2-20

O

Options
ACCUM - ANF, 10-107
CMBHAR - ANF, 10-107
DEVIA - ANF, 10-107
GEMPIE - PCL, 9-81
SEEHAR - FNP, 10-108
SLTOHT - PCL, 9-81
TABLO - NWT, 9-92, 9-94, 9-95
Ordinary Coefficient, 2-19, 2-20
Ordinary part
Of TAB file, 2-19
Overall Accuracy Summary
Faces (smiling or frowning), 9-80
On Solution file, 9-89

P

Postsim Coefficient, 2-19, 2-20
Postsim part
Of TAB file, 2-19, 2-20
Postsim pass, 2-25
Postsim pass 2
Assertions may be done, 9-85
Writes to terminal may be done, 9-85

Postsim processing, 2-10
  Suppressing, 2-24
Postsim section
  Of TAB file, 2-10
Postsim Set, 2-20
Postsim sets/subsets
  Not on Equations file, 2-29
Post-simulation values of Coefficient
  Meaning, 2-5
Preliminary pass, 2-25
  Assertions may be done, 9-85
  Writes to terminal may be done, 9-85
Program options. *See* Options
Programs
  Common features. *See* chapter 5 of GPD-1

Q

Quotas. *See* chapter 16 of GPD-3

R

Rae, Allan, 7-57
Range of values, 8-72
Ranking sets, 3-31
Rate% shock statement, 9-97
Read
  IfHeaderExists, 8-71
Real homogeneity. *See* chapter 13 of GPD-4
Reallocation
  Memory, 5-48
Relative file names. *See* chapter 5 of GPD-1
Report
  Newton error, 9-94
Reserved words, 8-72
Rimmer, Maureen, 4-40, 4-44
Rose, Steven, 6-56
RunDynam. *See* chapter 2 of GPD-4
  Working with Release 9 of GEMPACK, 12-111
RunDynam spreadsheet jobs
  Suppressing arithmetic errors, 10-107
RunGEM. *See* chapter 2 of GPD-4
  Last column in levels results, 9-82
  MIN file may not be needed, 11-109
RunGTAP
  Working with Release 9 of GEMPACK, 12-111
RunMONASH
  Working with Release 9 of GEMPACK, 12-111
RWHAR
  Documentation. *See* chapter 11 of GPD-4
RWSOL
  Documentation. *See* chapter 11 of GPD-4

S

SAGEM
  Documentation. *See* chapter 10 of GPD-3
SEEHAR
  Documentation. *See* chapter 4 of GPD-4
SEENV
  Documentation. *See* chapter 12 of GPD-4
Set and element labelling
  Documentation. *See* chapter 5 of GPD-4. *See* chapter 5 of GPD-4
Set element names, 8-72
Set Equality, 3-36
  Restricted to non-intertemporal sets (Release 8), 3-36
  Subset statements generated, 3-36
Set Mappings
  On updated data files, 8-73
Set size
  Formula for calculating, 3-32
Sets
  Equal. *See* Set Equality
  Postsim not on Equations file, 2-29
  Ranking, 3-31
Several subintervals, 9-103
Share_memory, 4-40
Shock
  Slice of array, 9-100
Shocks
  Specifying. *See* chapter 5 of GPD-3
Short file names. *See* chapter 5 of GPD-1
SLC file, 2-28
  May contain updated Coefficient values, 9-83
SLC files
  Values of postsim Coefficients, 9-83
Slice
  In shock statements, 9-100
  Of an array, 9-100
Slice dimension, 9-102
Slice element, 9-102
SLTOHT
  Command line, 10-106
  Documentation. *See* chapters 8 and 9 of GPD-4
  Subtotal, 6-55
Solution files
  Documentation. *See* chapter 8 of GPD-3
  Now Header Array files, 9-102
Solution methods
  Documentation. *See* chapter 7 of GPD-3
Source-code GEMPACK
  Installing on Windows PCs. *See* GPD-6
Source-code Version. *See* chapter 1 of GPD-1
Spaces
  In file names, 10-105
SSA. *See* chapter 2 of GPD-4
SSE
  File Qualifier in TAB files, 9-79
SSE output
  From GEMSIM and TG-programs, 9-79
SSL solutions, 9-102
Standard file suffixes. *See* chapter 5 of GPD-1
STI file. *See* Stored-input file
Stored-input files. *See* chapter 5 of GPD-1
  On command line. *See* chapter 5 of GPD-1
Subintervals, 9-103
Subsets
  Postsim not on Equations file, 2-29
Subtotal
  Accurate run, 7-60
  Documentation. *See* chapter 11 of GPD-3
  SLTOHT, 6-55
Subtotals results
  When SSL is selected, 9-102
Subtotals with complementarities
  Examples, 7-61
SUMEQ
  Documentation. *See* section 13.2 of GPD-4
SUMHAR
  Documentation. *See* chapter 4 of GPD-4
Swap statements, 9-89

Systematic Sensitivity Analysis. *See* chapter 2 of GPD-4

T

TAB file. *See* TABLO Input file
  Normal part, 2-20
  Postsim part, 2-20
TAB files
  With Postsim additions, 1-3
TABLO
  Command line, 10-106
TABLO-generated programs
  Compile and link on Windows PCs. *See* section 6.3 of GPD-6
  Compiling and linking. *See* section 1.2 of GPD-2
  Documentation. *See* GPD-3
  Executable image. *See* section 1.2 of GPD-2
TABmate. *See* chapter 2 of GPD-4
Tariff-rate quotas. *See* chapter 16 of GPD-3
Tax
  Ad valorem rate, 9-97
  Connection between power and *ad valorem* rate, 9-97
  Power, 9-97
Temporary copies of files
  Deleting, 10-107
TERM model, 1-3, 2-11
Text data files
  Documentation. *See* chapter 6 of GPD-4
TEXTBI
  Documentation. *See* chapter 14 of GPD-4
TG program. *See* TABLO-generated program
TGSVRAMNZ
  Reallocates MMNZ etc, 5-48
Tien Pham Duc, 1-4
Transferring data to/from GAMS. *See* chapter 16 of GPD-4
Transpose
  Of LHS Matrix, 5-51
Tsigas, Marinos, 1-4

U

UDC file, 2-29
  Written on Postsim Pass 1, 9-83
Unix machines
  Installing GEMPACK, 1-4
Unix/Command prompt. *See* section 2.3 of GPD-1
Unlimited Executable-image Version. *See* chapter 1 of GPD-1
Updated values of Coefficient
  Meaning, 2-5
UPPER_BOUND
  In Default statement in TAB files, 8-72
USAGE model, 4-40, 4-44

V

Valenzuela, Ernesto, 9-97
van Meijl, Hans, 6-55
van Tongeren, Frank, 1-4, 7-57
Versions of GEMPACK, 1-1. *See* chapter 1 of GPD-1
ViewHAR. *See* chapter 2 of GPD-4
ViewSOL. *See* chapter 2 of GPD-4
  Last column in levels results, 9-81
Virtual memory
  Using it can be very slow, 4-40

W

Windows PCs
  Installing GEMPACK, 1-4
WinGEM. *See* chapter 2 of GPD-4
Winners, 3-31
  Example with ORANIG, 3-32
Wittwer, Glyn, 9-82
World Wide Web, 1-3
Writes to terminal
  On Postsim pass 2, 9-85
  On preliminary pass, 9-85

X

XDISPLAY, 2-24
XPOSTSIM statements, 2-24
XWRITE, 2-24

Z

Zlatev Pivots, 5-52