GEMPACK USER DOCUMENTATION

Release 8.0

GPD-3 Simulation Reference :
GEMSIM, TABLO-generated Programs and SAGEM

Simulation Reference :

GEMSIM, TABLO-generated Programs and SAGEM

GEMPACK Document No. GPD-3

W. J. Harrison

K. R. Pearson

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

Second edition
October 2002

April 2007, reproduced in PDF with navigation links. Pagination may differ slightly from the October 2002 printing.

© Copyright 1985-2002. The Impact Project and KPSOFT.

Second edition
October 2002

ISSN 1030-2514

ISBN 0-7326-1522-4

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. GEMPACK provides software for calculating accurate solutions of an economic model, starting from an algebraic representation of the model.

Simulations to solve for the endogenous variables given the exogenous variables and the shocks to these variables, are run in GEMPACK using one of the following programs:

- the program GEMSIM,
- the TABLO-generated program written by TABLO from the TABLO Input file for the model, or
- the program SAGEM.

This document is a complete reference for running simulations. Details are given of the syntax used in Command files and the methods used in solving the equations of the model.

If you have no prior knowledge of GEMPACK, we recommend you start with the GEMPACK document GPD-1 *An Introduction to GEMPACK*. In this document GPD-3, we assume you are familiar with the material covered in GPD-1.

Document Attributes

Name	: Simulation Reference : GEMSIM, TABLO-generated Programs and SAGEM
Audience	: CGE Modellers
Identifier	: GPD-3

History :	Date	Author(s)	Comment
	October 2000	Jill Harrison and Ken Pearson	First Edition (Release 7.0)
	October 2002	Jill Harrison and Ken Pearson	Second Edition (Release 8.0)

In the Release 6.0 documentation, GEMPACK document GPD-3 was "How to Create and Modify GEMPACK Header Array Files Using the Program MODHAR". This material is now in Chapter 3 of GPD-4.

Table of Contents

1. INTRODUCTION 1-1

2. COMMAND FILES 2-3

2.1 Simulations using GEMSIM or TABLO-generated Programs 2-3

2.2 Data Manipulation using GEMSIM or TABLO-generated Programs 2-4

2.3 SAGEM Simulations 2-5

2.4 File Names 2-5
2.4.1 Naming the Command File 2-5
2.4.2 File Names Containing Spaces 2-5

2.5 Using the Command File Stem for Names of Other Output Files 2-6
2.5.1 Default: Solution File Stem = Command File Stem 2-7
2.5.2 Default: Log File Stem = Command File Stem 2-7
2.5.3 Default: Display File Stem = Command File Stem 2-7
2.5.4 Using <Cmf> in Command Files 2-8

2.6 Log Files and Reporting CPU Time 2-8

2.7 General Points about Command File Statements 2-8

2.8 Eliminating Syntax Errors in GEMPACK Command Files 2-10

3. TABLO INPUT FILES AND AUXILIARY FILES 3-11

3.1 GEMSIM and GEMSIM Auxiliary Files 3-11

3.2 TABLO-generated Programs and Auxiliary Files 3-12
3.2.1 Windows PCs (Except for Version 5.50 of LF95) 3-12
3.2.2 Unix Machines and Windows PCs using Version 5.50 of LF95 3-12

3.3 How are the Names of the Auxiliary Files Determined? 3-13

3.4 Check that Auxiliary Files are Correct 3-14
3.4.1 Model Information (MIN) File 3-14
3.4.2 Compatibility of Various Files 3-14

3.5 Carrying out Simulations on Other Machines 3-14
3.5.1 Copying TABLO-generated Programs to Other Computers (Same Operating System) 3-15
3.5.2 GEMPACK Licence May Be Required 3-15

4. DATA FILES, UPDATED DATA FILES AND DISPLAY FILES 4-17

4.1 Data Files and Logical Files 4-17
4.1.1 Input and Output Data files 4-18

4.2 Updated Data Files 4-20
4.2.1 Naming Updated Files 4-20
4.2.2 Updated Data Read from the Terminal 4-21

4.2.3 Intermediate Data Files 4-21
4.2.4 Long Names on Updated Data Files 4-21

4.3 Display Files 4-21
4.3.1 Options for Display Files 4-22

4.4 Checking Set and Element Information when Reading Data 4-22
4.4.1 Check-on-Read Problems with RunGEM, RunDynam, RunMONASH and RunGTAP 4-25
4.4.2 Checks When Reading Set Element Names from a File 4-25
4.4.3 Checks on Run-time Elements used in TABLO Input File 4-25

4.5 Names of Intermediate Data Files 4-26
4.5.1 Default Names for Intermediate Files When Using a Command File 4-27

5. CHOOSING THE CLOSURE AND SHOCKS 5-29

5.1 Specifying the Closure 5-29
5.1.1 Miniature ORANI Model 5-30

5.2 Specifying the Closure via a Command File 5-31
5.2.1 Specifying the Closure from Scratch 5-31
5.2.2 Using a Saved Closure 5-32
5.2.3 Modifying a Saved Closure 5-33
5.2.4 Order of Command File Statements Can Be Important 5-33
5.2.5 Closure Specified Using Data Files 5-34
5.2.6 List of Exogenous/Endogenous Variables when Closure is Invalid 5-35
5.2.7 Initial Closure Check and Singular Matrices 5-35

5.3 Component Numbers for a Variable 5-37
5.3.1 Examples of Component Numbers 5-37

5.4 Choosing Other Sets of Variables 5-38
5.4.1 Choosing Sets of Variables via a Command File 5-38
5.4.2 Example of Choosing Sets in SAGEM 5-40
5.4.3 Choosing Sets of Variables Interactively 5-42

5.5 Specifying the Shocks 5-43
5.5.1 Specifying the Shocks via a Command File 5-43
5.5.2 Using a Shock File 5-44
5.5.3 Components Must Be In Increasing Order in Most Cases 5-47
5.5.4 Additional and Target Shock (Ashock or Tshock) Statements for Dynamic Models 5-48
5.5.5 Shocking a Variable with No Components Exogenous 5-49

5.6 FINAL_LEVEL Statements : Alternatives to Shock Statements 5-50

5.7 CHANGE or PERCENT_CHANGE Shock Statements 5-52

5.8 Shock Files – Fine Print 5-53
5.8.1 All Components Shocked and "Select From File" Cases 5-53
5.8.2 Shocking Some Components (Not "Select From File") 5-56

5.9 Checking the Closure and Shocks 5-58

5.10 Levels Variable Names can be Used in Command Files 5-59

5.11 Clarification Concerning Shocks on a Command File 5-59
5.11.1 Shock Statement When Only Some Components are Exogenous 5-59
5.11.2 No "Shock" Statements on a Command File 5-59

5.12 Specifying the Shocks Interactively 5-60

6. ACTIONS IN GEMSIM AND TABLO-GENERATED PROGRAMS 6-61

6.1 Possible Actions in GEMSIM and TABLO-generated Programs 6-61
6.1.1 Multi-Step Simulations with Economic Models 6-61
6.1.2 Creating an Equations File 6-61
6.1.3 Other Actions (Writes, Displays, Assertions, Range Tests and Transfers) 6-62
6.1.4 Data Manipulation 6-62
6.1.5 "Extra" (TABLO-like) Actions 6-63
6.1.6 Checking the Closure and Shocks 6-63
6.1.7 Controlling Whether and How the Actions are Carried Out 6-63
6.1.8 Some READs and FORMULAS May be Omitted 6-63
6.1.9 Writes and Displays at All Steps of a Multi-step Simulation 6-64
6.1.10 Echoing Activity 6-64
6.1.11 Writes to the Terminal 6-64

6.2 How These Programs Carry Out Multi-step Simulations 6-65
6.2.1 Processing the Closure and Shocks in GEMSIM or TABLO-generated Programs 6-67
6.2.2 Results of a 4-step Simulation Looked at in Detail 6-68
6.2.3 Are the Linear Equations in the TAB File Satisfied by the Accurate Results? 6-69

6.3 Assertions 6-72

6.4 Range Tests 6-73
6.4.1 How to Specify the Allowed Range for Values of a Coefficient or Levels Variable 6-74
6.4.2 Tests Carried Out 6-75
6.4.3 An Example 6-76
6.4.4 Associated Statements in Command Files 6-78

6.5 Transfer Statements 6-79

6.6 TABLO-like Statements in Command Files 6-79
6.6.1 TABLO-like Checks of Extra Statements 6-80
6.6.2 Qualifiers 6-81
6.6.3 Other Points 6-81

6.7 Coefficients Are Fully Initialised by Default 6-82

6.8 How Accurate Are Arithmetic Calculations? 6-83
6.8.1 Example 1 – GTAPVIEW 6-83
6.8.2 Example 2 – Checking Balance of ORANIG Data 6-84

7. MULTI-STEP SOLUTION METHODS 7-87

7.1 What Method and How Many Steps to Use ? 7-87
7.1.1 Example – GTAP Liberalization Simulation 7-88
7.1.2 Command File Statements for Method and Steps 7-90

7.2 Extrapolation Accuracy Summaries and Files 7-91
7.2.1 Accuracy Estimates and Summaries For Variables 7-91
7.2.2 Accuracy Estimates and Summaries For Updated Data 7-92
7.2.3 Extrapolation Accuracy Files 7-93
7.2.4 Command File Statements Affecting Extrapolation Accuracy Summaries and Files 7-94

7.3 Splitting a Simulation into Several Subintervals 7-96
7.3.1 Overall Accuracy Summary When More Than One Subinterval 7-98

7.4 Automatic Accuracy for Simulations 7-98
7.4.1 Specifying Which Solutions to Use 7-99
7.4.2 Incompatibilities 7-100
7.4.3 Automatic Accuracy with Models which Read Data from the Terminal 7-100
7.4.4 Adaptive Stepsize Method 7-100
7.4.5 Stopping if Subintervals Become Too Small 7-100
7.4.6 Controlling Which Values are Tested 7-101
7.4.7 Checking Progress During a Run 7-101

7.5 Newton's Method for Levels Models 7-102
7.5.1 Newton's Method and ORANIF 7-103
7.5.2 Convergence 7-103
7.5.3 General advice 7-103
7.5.4 Newton's Method Is Not Reliable with Mixed Models 7-104

7.6 Homotopy Methods and Levels Equations 7-105
7.6.1 Example 1 – Solving a System of Levels Equations 7-105
7.6.2 Example 2 – Finding an Intertemporal Data Set for an Intertemporal Model 7-106
7.6.3 Example 3 – Use When Adding Behaviour to a Model 7-107
7.6.4 Formal Documentation about ADD_HOMOTOPY 7-108
7.6.5 ADD_HOMOTOPY and NOT_ADD_HOMOTOPY Default Statements 7-109

7.7 Options for Saving Solution and Updated Data Files 7-109
7.7.1 Saving Solutions or Updated Data After Separate Multi-step Calculations 7-109
7.7.2 Saving Updated Values from All FORMULA(INITIAL)s 7-110

7.8 Solutions Report Perturbations of the Initial Values 7-111
7.8.1 Change Differentiation Keeps a Constant Error 7-112
7.8.2 How Can Homotopies Help? 7-113

8. SOLUTION FILES 8-115

8.1 Command File Statements Related to Solution Files 8-115
8.1.1 SAGEM Only Command File Statements 8-116

8.2 Contents of Solution Files 8-116
8.2.1 TABLO File and TABLO STI File Stored on Solution File 8-117
8.2.2 Command File Stored on Solution file 8-117

8.3 Levels Results 8-117

8.4 Solution Coefficients (SLC) Files 8-118
8.4.1 Solution and SLC Files are Complete Record of the Simulation 8-118
8.4.2 System-Initiated Coefficients on SLC File 8-118

8.5 Updated and Average Coefficient Values (UDC and AVC) Files 8-119
8.5.1 How Values on UDC Files are Calculated 8-119
8.5.2 Values on UDC Files May Differ From Values Calculated from the Updated Data 8-119
8.5.3 Are Values on UDC Files Different From Values Shown in the Updated Data? 8-120

8.6 Coefficient Values (CVL) Files from Data Manipulation TAB Files for AnalyseGE 8-120

8.7 Coefficient Values (CVL) Files For Analysing Why a Simulation Crashes 8-121

9. EQUATIONS FILES AND BCV AND LU FILES 9-123

9.1 Equations Files 9-123
9.1.1 Using an Equations File in SAGEM 9-123

9.1.2 Differences in Step 1 Depending on Whether or Not Equations File is Saved 9-124
9.1.3 Model Name, Version and Identifier 9-124

9.2 Base Coefficient Values (BCV) Files 9-125
9.2.1 Starting from Existing Equations and BCV Files Not Recommended 9-125

9.3 LU Files 9-126

10. SEVERAL SIMULTANEOUS JOHANSEN SIMULATIONS VIA SAGEM 10-127

10.1 The Solution Matrix 10-127
10.1.1 Individual Column Results 10-128
10.1.2 Cumulative or Row Totals Results 10-128
10.1.3 Subtotal Solutions 10-129
10.1.4 Contents of a SAGEM Solution File 10-129
10.1.5 Viewing the Results of a Simulation 10-129

10.2 Subtotals 10-130
10.2.1 First Method using SAGEM 10-130
10.2.2 Second Method using GEMPIE 10-131
10.2.3 Subtotals and Sets of Shocks in General 10-131
10.2.4 Subtotal Descriptions 10-131
10.2.5 Viewing Subtotals or Individual Column Results using GEMPIE or ViewSOL 10-132

10.3 No Individual Column Results from Multi-Step Simulations 10-132

11. SUBTOTALS VIA GEMSIM OR TABLO-GENERATED PROGRAMS 11-133

11.1 Subtotals from GEMSIM and TABLO-generated Programs 11-133

11.2 Meaning of Subtotals Results 11-133

11.3 Subtotal Example 11-134

11.4 More Substantial Examples 11-135

11.5 Processing Subtotal Results 11-135

11.6 Subtotals Results Depend on the Path 11-135
11.6.1 Example Showing How the Path Can Affect Subtotals Results 11-136

12. SOLVING MODELS AND SIMULATION TIME 12-137

12.1 How GEMPACK Programs Solve Linear Equations 12-137
12.1.1 Sparse-Linear-Equation Solving Routines MA48 and MA28 12-138
12.1.2 Which to Use? MA48 or MA28 12-138

12.2 Gragg's Method and the Midpoint Method 12-139

12.3 Reusing Pivots 12-140

12.4 Time Taken for the Different Steps 12-141
12.4.1 Total Time for Multi-step Compared to 1-step 12-141
12.4.2 Using Existing Files to Speed Up Step 1 12-142

12.5 Ignoring / Keeping Zero Coefficients 12-143

12.6 Equation Solving Problems 12-144
12.6.1 Warnings About Equations Not Being Satisfied Very Accurately 12-144
12.6.2 Warnings About Variation Between Passes in Endogenous Results 12-145
12.6.3 Gragg and Midpoint Are Not Suitable in Some Cases 12-146
12.6.4 Linearization Used Can Affect Accuracy 12-146
12.6.5 Example of Increasingly Large Oscillations 12-148
12.6.6 Scaling Equations 12-148

12.7 Work Files 12-148

13. MEMORY MANAGEMENT 13-149

13.1 Automatic Memory Allocation 13-149

13.2 Preliminary Pass 13-149

13.3 Repetitions if Memory Allocation is Too Small 13-150

13.4 Reporting Memory Used by TABLO-generated Programs and GEMSIM 13-151

14. OPTIONS FOR GEMSIM AND TABLO-GENERATED PROGRAMS 14-153

14.1 Options Affecting Simulations 14-154

14.2 Options Affecting Extrapolation Accuracy Summaries and Files 14-154

14.3 Saving Updated Values from All FORMULA(INITIAL)s 14-154

14.4 Options Affecting the Actions Carried Out 14-154

14.5 Options Affecting How Writes and Displays are Carried Out 14-155

14.6 Options Affecting CPU and Activity Reporting 14-156

14.7 Special Options for SAGEM 14-156
14.7.1 SAGEM Options 14-156

15. RUN-TIME ERRORS 15-159

15.1 Simulation Fails Because of a Singular LHS Matrix 15-160
15.1.1 Trying to Fix a Singular Matrix Problem 15-161

15.2 Structurally Singular Matrices 15-161
15.2.1 Solving Modified Equations when LHS Matrix is Structurally Singular 15-163

15.3 Finding Where an Arithmetic Error Occurs and Rectifying It 15-164

15.4 Division by Zero Errors 15-165

15.5 Error Trapping for Invalid Powers, Logs etc 15-165

15.6 Overflow and Other Arithmetic Errors 15-166
15.6.1 Other Sorts of Arithmetic Errors 15-166

16. SIMULATIONS FOR MODELS WITH COMPLEMENTARITIES 16-167

16.1 The Basic Ideas 16-168
16.1.1 If Only We Knew the Post-Simulation States 16-168
16.1.2 Finding the Post-Simulation States 16-168
16.1.3 Combining the Two Ideas 16-169

16.2 States of the Complementarity 16-172
16.2.1 Complementarities with Only One Bound 16-172

16.3 Writing the TABLO Code for Complementarities 16-174
16.3.1 Import Quota Example 16-174
16.3.2 Example: MOIQ (Miniature ORANI with Import Quotas) 16-174
16.3.3 Specifying Bounds for the Complementarity Variable 16-176
16.3.4 Scale the Complementarity Expression if Possible 16-176
16.3.5 Levels Variables are Required 16-177
16.3.6 Complementarities are Related to MAX and MIN 16-177
16.3.7 Equivalent Ways of Expressing Complementarities with a Single Bound 16-178

16.4 Hands-on Examples with MOIQ 16-180
16.4.1 Decreasing a Quota Volume – Command File MOIQ1C.CMF 16-180
16.4.2 Decreasing both Quota Volumes – Command file MOIQ1D.CMF 16-182
16.4.3 No Approximate Run – Command file MOIQ1CX.CMF 16-182

16.5 Other Features of Complementarity Simulations 16-182
16.5.1 Closure and Shocks 16-182
16.5.2 Running TABLO and Simulations 16-183
16.5.3 Program Reports State Changes 16-183
16.5.4 Program Checks States and Bounds At End of Run 16-183
16.5.5 Checking the Simulation Results 16-184
16.5.6 Omitting the Accurate Run 16-184

16.6 Command File Statements for Complementarity Simulations 16-184

16.7 Technical Details about Complementarity Simulations 16-186
16.7.1 How the Closure and Shocks Change for the Accurate Simulation 16-186
16.7.2 Extra Variables Introduced with Each Complementarity. 16-187
16.7.3 Step May Be Redone During Approximate Run 16-189
16.7.4 Several Subintervals and Automatic Accuracy 16-189
16.7.5 Checks that the Pre- and Post-Simulation States are Accurate 16-190

16.8 Complementarity Examples 16-192
16.8.1 Example: MOIQ (Miniature ORANI with Import Quotas) 16-192
16.8.2 Example: MOTRQ (Miniature ORANI with Tariff-Rate Quotas) 16-192
16.8.3 Example: MOTR2 (Alternative Form of Miniature ORANI with Tariff-Rate Quotas) 16-197
16.8.4 Example: G5BTRQ (GTAP with Bilateral Tariff-Rate Quotas) 16-198
16.8.5 Alternative Form of the Complementarity – G5BTR2.TAB 16-202
16.8.6 Example: G5GTRQ (GTAP with Global and Bilateral TRQs) 16-203
16.8.7 Example: G94-XQ (GTAP94 with Bilateral Export Quotas) 16-205
16.8.8 Example: G94-IQ (GTAP94 with Bilateral Import Quotas) 16-208

16.9 Piecewise Linear Functions via Complementarities 16-210
16.9.1 Max and Min 16-210
16.9.2 Progressive Income Tax Schedule 16-212
16.9.3 Other Piecewise Linear Functions 16-214

16.10 Example: ORANIGRD (Ensuring Investment Stays Non-negative) 16-218
16.10.1 OGRD1.TAB and Negative Investment 16-218
16.10.2 OGRD2.TAB – A Complementarity to Ensure Non-negative Investment 16-218

17. NEW COMMAND FILE STATEMENTS 17-223

17.1 New Command File Statements for Release 8.0 17-223

17.2 New Command File Statements for Release 7.0 17-223

17.3 New Command File Statements for Release 6.0 17-224

17.4 New Command File Statements for Release 5.2 17-224

18. SUMMARY OF COMMAND FILE STATEMENTS 18-227

18.1 Command Files for GEMSIM and TABLO-generated Programs 18-227
18.1.1 Command File Statements 18-228
18.1.2 Complete Command File Example for GEMSIM and TABLO-generated Programs 18-243

18.2 Command Files for Running SAGEM 18-244
18.2.1 Complete Command File Example for SAGEM 18-249

19. REFERENCES 19-251

20. GEMPACK DOCUMENTS 20-253

21. INDEX 21-255

CHAPTER 1

1. Introduction

TABLO Input files can be written to implement an economic model. TABLO and TABLO Input files are described in GEMPACK document GPD-2 *TABLO Reference.*

Following implementation, one of the GEMPACK simulation programs, GEMSIM, a TABLO-generated program or SAGEM, is used to carry out a **simulation**. The aim of the simulation is to solve the equations of the model for the values of the endogenous variables, given the exogenous variables and the shocks applied to them.

TABLO Input files can also be written to carry out **data manipulation**. GEMSIM or a TABLO-generated program can be used to read in some data from data files, evaluate some formulas and then write some data to new data files.

The usual method of running a simulation or carrying out data-manipulation task is to use a **Command file,** as described in chapter 2 of GPD-1. While it is possible to run GEMPACK programs interactively at the Command prompt, we recommend that you never run a simulation or data-manipulation task interactively, but always use a Command file. A Command file gives a good brief record of what the simulation was about and can be used to rerun the simulation later if necessary.[1]

This document describes in detail simulations and data-manipulation tasks and provides a detailed reference for the statements used in a Command file to control these processes. Chapter 2 introduces some important features of Command files. Chapter 3 describes the connection between a Command file and the TABLO Input file. Chapter 4 deals with original and updated data files, and Display files. Chapter 5 describes how the closure and shocks are specified on a Command file. Chapter 6 describes the various possible actions apart from solving a model: these include writes, displays, assertions, range tests and transfers. TABLO-like statements can be included in Command files, as described in section 6.6. There is a summary of all Command file statements in Chapter 18. Chapter 17 contains a list of Command file statements which were new in recent releases of GEMPACK.

Chapter 7 describes how GEMSIM and TABLO-generated programs calculate accurate solutions of the underlying (usually nonlinear) equations of a model. Subtotals results can be calculated by GEMSIM or TABLO-generated programs as described in chapter 11. Chapter 12 gives some of the slightly more technical details about solving models.

Chapters 8 and 9 contain details about Solution, Equations and Base Coefficient Values files.

Chapter 10 gives details about how SAGEM can produce one or more Johansen solutions to a simulation.

Chapter 16 describes how complementarities can be modelled and how quotas and tariff rate quotas can be handled in GEMPACK.

Other topics covered include memory management (see chapter 13), options for GEMSIM and TABLO-generated programs (chapter 14) and run-time errors (chapter 15).

[1] Sometimes you may wish to use a Stored-input file but often the Stored-input file uses the option CMF to call a Command file.

CHAPTER 2

2. Command Files

2.1 Simulations using GEMSIM or TABLO-generated Programs

In section 2.8 of GEMPACK document GPD-1, there is an introductory description of the information needed to carry out a simulation on a model. The following topics cover much of the information needed.

- **TABLO Input File and Auxiliary Files**

When you build a model in GEMPACK, you write down the theory in the TABLO Input file for your model. When you run TABLO, you can choose either to produce output for GEMSIM, or to produce the Fortran program which we call the TABLO-generated program for your model.

The economic model is converted by TABLO into either

* a TABLO-generated program and associated Auxiliary files or
* GEMSIM Auxiliary files.

See chapter 3 for Command file statements about TABLO Input files and Auxiliary files.

- **Data Files and Updated Data Files**

Simulations start from an initial solution of the model stored on data files. See section 4.1 for Command file statements for data files.

After running the simulation, the percentage change and ordinary change variables are applied to the initial data files to produce the updated data files. See section 4.2 for statements about updated files.

- **Closure and Environment Files**

The closure of the model for a particular simulation specifies which variables are exogenous (that is, their values are given as shocks or are unchanged in levels) and which variables are endogenous (that is, the variables calculated when the model is solved). See chapter 5 for details about specifying a closure and the use of Environment files.

- **Shocks and Shock Files**

The shocks are applied to some of the exogenous variables in a simulation – see section 5.5 for details.

- **Multi-Step Solution Methods**

Multi-step solution methods are used in GEMSIM or TABLO-generated programs to solve the usually non-linear equations of the model. An introduction is given in section 2.13 of GPD-1.

Chapter 7 begins with detailed advice about what solution method to use.

For details about Command file statements for the method of solution, and number of steps, see section 7.1. Other topics covered in chapter 7 are how to specify the required accuracy, the use of subintervals, and automatic accuracy to improve the accuracy of the solution.

- **Solution Files**

These contain the results of the simulation (that is, the values of the endogenous variables). They also contain information about the simulation itself (for example, the shocks). Indeed, Solution files produced by Release 7.0 or later programs contain a copy of the Command file. See section 8.2 for details about what is stored on a Solution file and how to recover this information if necessary.

- **Verbal Description**

A verbal description (one or more lines of text) is required in the Command file for any simulation. See section 8.1 for details.

- **Names of Other Output Files**

See section 2.5 about various output details such as output file names, and ways to make editing your Command files more efficient, for example using <cmf> in Command files.

- **Other Actions**

Besides solving the model, TABLO-generated programs and GEMSIM can carry out other actions, including writes, displays, assertions, range tests and transfers. See chapter 6 for details.

- **Subtotals in Multi-step Simulations**

With Release 7.0 (or later) of GEMPACK, you can use subtotals to find the effect of just some of the shocks on the endogenous variables in the cumulative solution calculated using GEMSIM or a TABLO-generated program – see chapter 11.

- **TABLO-like Statements in Command files**

TABLO-like statements in Command files can also be used and are described in section 6.6.

- **Special Options**

Special options available for Command files (some of which can be chosen via the initial options menu of programs) are described in chapter 14.

2.2 Data Manipulation using GEMSIM or TABLO-generated Programs

Data-manipulation programs were introduced and an example given in section 4.4 of GPD-1.

- **Data-manipulation TABLO Input Files and Auxiliary Files**

The task is to read in some data from data files, use formulas to manipulate the data in some way and write the new data to new data files. This task is written down in a TABLO Input file which contains reads, formulas, writes and displays, but **no equations**. The TABLO Input file is converted by TABLO into computer files containing the TABLO-generated program or GEMSIM Auxiliary files.

See chapter 3 for Command file statements about TABLO Input files and Auxiliary files.

- **Data Files and New Output Data Files**

The main statements in the Command files used for data manipulation are the statements specifying the names of the old files containing data read in and new data files written after the new data has been calculated. See section 4.1 for Command file statements for data files.

See also section 2.5 about ways of writing the names of output files.

- **Actions**

Data-manipulation programs can carry out various actions, including writes, displays, assertions, range tests and transfers. See chapter 6 for details.

- **TABLO-like Statements in Command files**

TABLO-like statements in Command files can also be used and are described in section 6.6.

- **Special Options**

Special options available for Command files (some of which can be chosen via the initial options menu of programs) are described in chapter 14.

2.3 SAGEM Simulations

SAGEM is a GEMPACK program which carries out Johansen (one-step) simulations on an economic model. SAGEM simulations are introduced in section 2.12 of GPD-1. See chapter 10 for details about Johansen solutions and how to calculate several simultaneous solutions using SAGEM.

- **Equations File**

The Equations file used as a starting point by SAGEM must have been created by using either GEMSIM or a TABLO-generated program – see section 9.1.1 of this document and also section 2.12 of GPD-1. The Equations file contains the matrix of equations calculated from the equations in the economic model and the initial data.

- **Closure and Environment Files**

For details about specifying a closure and the use of Environment files, see chapter 5.

- **Shocks and Shock Files**

The shocks are applied to some of the exogenous variables in a simulation – see section 5.5 for details.

- **Solution Files**

See chapter 10 for details about what is stored on a SAGEM Solution file. In particular, SAGEM can produce individual column and subtotals solutions in addition to the cumulative solution.

- **Verbal Description**

See section 8.1 about the verbal description.

2.4 File Names

The names of actual files on your computer can cause various problems. See GEMPACK document GPD-1 section 5.9 for general points which apply to all file names.

2.4.1 Naming the Command File

The name of the Command file can be used to remind you about the simulation which it carries out. Since the simulation relies on the underlying model, the closure and also the shocks, each of these could form part of the Command file name.

For example, if you are running the ORANIG model, using a shortrun closure, and giving a wage rise of 10 percent, you could call your Command file ORANIG_SR_Wage10.CMF or OGSRWage.CMF, if you prefer shorter names.

See section 5.9 of GPD-1 for information about the file and directory names allowed on different machines. See section 2.5 about how to use this Command file name as a basis for the names of other output files such as the Solution file.

2.4.2 File Names Containing Spaces

On Windows PCs, file names can contain spaces[2]. This is usually possible on Unix machines – consult your GEMPACK Manager to find out. Details about this can be found in section 5.9 of GPD-1.

To specify a file name containing spaces in a Command file, enclose the name in double quotes **" "** as in, for example,

```
Solution file = "c:\my  sj\sjlb" ;
Auxiliary files = "c:\my  model\my  model" ;
```

Trailing spaces are always removed. For example

```
Solution file = "c:\my  sj\sjlb    " ;
```

[2] File names cannot contain spaces if you are using an old program compiled using the Fortran77 compiler F77L3.

is treated as if you had written

```
Solution file = "c:\my  sj\sjlb" ;
```

2.5 Using the Command File Stem for Names of Other Output Files

When using Command files, many experienced users have adopted the convention of calling the Solution file the same name as the Command file. For example, if the Command file is SJLB.CMF, the Solution file is often called SJLB.SL4 . If you change the closure or shocks, it is natural to save the Command file to a new name. To preserve the Command file /Solution file name-link, you then also need to edit the Command file to change the Solution file name if the Solution file name were included in the Command file.

Similarly, it is good practice to choose names for the updated data and other output files (such as LOG files and Display files) which are similar to, or identical to, the name of the Command file. For example, in the Command file SJLB.CMF for a simulation with the Stylized Johansen model (which has only one logical data file called IODATA – see chapters 2 and 3 of GPD-1), the updated version of IODATA would be called SJLB.UPD and the LOG file called SJLB.LOG.

In Release 5.2 or earlier of GEMPACK, in the Command file SJLB.CMF for a simulation with Stylized Johansen, you would usually have seen the lines

```
solution file = sjlb ;
updated file iodata = sjlb.upd ;
log file = sjlb.log ;
```

Since Release 6.0, you can leave these names to be inferred and it would be better practice to replace those lines by

```
updated file iodata = <cmf>.upd ;
log file = yes ;
```

- Notice first that there is no "solution file = … ;" statement. This is because if you don't include a line "solution file = … ;", the Solution file name is taken from that of the Command file. [So if the Command file is called SJLB.CMF the Solution file name is automatically taken to be SJLB, followed by the standard Solution file suffix which is usually .SL4.]
- The <cmf> in the statement **`updated file iodata = <cmf>.upd ;`** is replaced by the pre-suffix name of the Command file – that is, the name of the Command file which comes before the suffix .CMF[3]
- The statement **`log file = yes ;`** indicates that you want a LOG file and that you want its name to be the same as that of the Command file except that its suffix will be ".log".

Note that neither of the two lines above needs to be changed when you copy this file as the basis of a new simulation.

Note that the defaults above only apply when the name of the Command file has suffix ".CMF" (any case – upper ".CMF", lower ".cmf" or mixed).

Along these lines we have introduced a similar default for the name of Display files (see section 4.3), namely

> if you don't include a line "display file = … ;", the Display file name is taken from that of the Command file and is given suffix ".dis" (again provided the Command file has suffix ".CMF" as above).

Note that if you have several updated data files you can use <cmf> as just part of the name – for example in a model with two logical files DATA1 and DATA2 which are updated you might include lines

```
updated file DATA1 = <cmf>1.upd ;
updated file DATA2 = <cmf>2.upd ;
```

---

[3] Often GEMPACK documents use the notation <….> as something you supply, for example, <file-name> where we want you to supply the filename you are using, and then you type in your own file name but do not type in the < or > symbols. Here you need to type into your Command file the string "<cmf>" and GEMSIM, the TABLO-generated program or SAGEM replaces the string "<cmf>" for you at run time.

to base both names on the Command file name but adding the extra characters 1 or 2 to distinguish between them. [If you do this, you need to be careful that names don't exceed the 8 characters allowed under DOS if you are using Lahey Fortran F77L3 – see section 5.9 of GPD-1.][4]

Another way of thinking of these conventions is as follows. If, for example, you have a Command file called XXX.CMF, GEMSIM, TABLO-generated programs or SAGEM will look in your Command file for a statement **`solution file = <file_name> ;`** If this statement is not found, the software will assume you want to use the stem of the Command file name XXX (minus the suffix .CMF) as the Solution file name and the program will proceed as if you had the statement
`solution file = XXX ;` in your Command file. Then if you change your closure or shocks and save the command file to a new name YYY.CMF, when you run your simulation the new Solution file will be called YYY.SL4 without your having to edit the Solution name in the *solution file = … ;* statement.

The defaults described here apply to Command files for GEMSIM, TABLO-generated programs and SAGEM.

We set out these defaults more formally below.

Command File Stem

We refer to the part of the Command file name without the suffix .CMF as the **Command file stem**.

This only applies to Command files which end in the suffix **.CMF** or **.cmf** (any case is allowed).[5] For example, for the file SJLB.CMF, the Command file stem is SJLB. Similarly the Solution file stem is the Solution file name minus the required suffix .SL4.

2.5.1 Default: Solution File Stem = Command File Stem

If the statement **`solution file = <file_name> ;`** is not present, the Solution file name is taken to be the stem of the Command file name, provided the Command file ends with suffix *.cmf* or *.CMF* (any case is allowed).

As in the example above this means that if the Command file name is ***sjlb.cmf***, the Solution file name is the default name ***sjlb.sl4*** unless there is a *Solution file = <file_name> ;* statement is in your CMF file to give your Solution file some other name.

In addition the name of the Command file is added by the program to the Verbal description so that it is clear which Command file was being used for this simulation

2.5.2 Default: Log File Stem = Command File Stem

If you put a statement **`Log file|only = yes ;`** in your Command file the log file name will be given the same stem as the Command file stem followed by the suffix **.log** (provided that the Command file name ends with suffix *.cmf* or *.CMF*). [Consult section 5.3.3 of GPD-1 to see the difference between "log file = yes ;" and "log only = yes ;".]

For example if you put the statement *log file = yes ;* in the Command file ***sjlb.cmf*** the log file would be called ***sjlb.log***

2.5.3 Default: Display File Stem = Command File Stem

If the statement *display file = <file_name> ;* is not present, the Display file name uses the stem of the Command file name, followed by the suffix **.dis** (provided the Command file ends with suffix *.CMF* as before).

So, if there are displays in your TABLO Input file but there is no *display file = ..;* statement in the Command file ***xxx.cmf***, the display file will be called ***xxx.dis***.

---

[4] See the GTAPVIEW example in section 4.1.1 for another example.

[5] The suffix .CMF is not required by GEMPACK (see section 5.8.2 of GPD-1). Thus you could use a Command file called SJLB.XYZ if you wished. However we strongly advise you to always use suffix .CMF for Command files so that you can take advantage of the defaults described in this section.

2.5.4 Using <Cmf> in Command Files

In writing Command files, you can use the shorthand notation ***<cmf>*** to stand for the Command file stem (again provided the CMF file name ends with *.CMF* or *.cmf*). For example, the following statements could be included in your command file:

```
equations file = <cmf> ;
updated file iodata = <cmf>99.upd ;
```

If your Command file was called FRCAST.CMF, this is equivalent to

```
equation files = FRCAST ;
updated file iodata = FRCAST99.upd ;
```

but if at a later stage you renamed your Command file to DEV.CMF, the statements become equivalent to

```
equations file = DEV ;
updated file iodata = DEV99.upd ;
```

2.6 Log Files and Reporting CPU Time

You can ask for a Log file by including a statement of the form

log file|only = … ;

in your Command file. If you include "log file =", output goes to the terminal and also to the log file. If you include "log only =", output only goes to the log file (once the program has started running and processed this statement). See section 2.5.2 for details about the statements "log file|only = yes;".

If you include the statement[6]

CPU = yes ;

in your Command file, GEMSIM, SAGEM or the TABLO-generated program will report CPU time (that is, processing time) for various parts of the code (for example, the time taken for all updates in each step). However, on some machines this may just report CPU times as zero, which means that CPU reporting is not available on this machine.

2.7 General Points about Command File Statements

This section describes various points that apply to all Command file statements throughout this document.

- LINES: Lines can be up to 500 characters long[7].
  Input is free-form. In particular, Command file statements can extend over more than one line, such as
  ```
  exogenous   pfac    xcom   xfacin 1-4 6-19
      yfacin z ;
  ```
- CASE: You can use upper or lower case. Command files are case-independent. For example, it doesn't matter if your command is
  'Equations file = … ' or 'equations FILE = … '.
- ALTERNATIVES: The notation "|" indicates alternatives in Command file syntax. For example,
  ```
  log file = yes|no ;
  ```
  means that you can use either "yes" or "no" as in
  ```
  log file = yes ;
  ```
  or `log file = no ;`
- USER TEXT: The notation < > shows user selected text. For example, you must select a filename to replace <file_name> in :

---

[6] Alternatively, select option CPU – see section 14.6.

[7] For Release 7.0, lines were limited to 120 characters.

```
equations file = <file_name> ;
```
You should not type in "<" or ">".

- ABBREVIATIONS: All other text in Command file statements is required, but words in a Command file can be abbreviated once they become unique. For example,
  `sol fil = sjlb ;` is an acceptable abbreviation for
  `solution file = sjlb ;` but
  `s file = sjlb;` is not accepted since 's' is too short to distinguish between possible first words such as *`solution`*, *`swap`*, *`shock`*, for example.

  However you cannot abbreviate user-selected input such as the names of variables or *`%macro`*.

- OPTIONAL TEXT: Square brackets **[ ]** indicates **optional** user-selected text.

- DEFAULTS: If a default is specified, this is the value if no such command is given. When a default is specified, the default action will take place if you omit the command. For example, if you have no command '`Harwell parameter = …`', the parameter will be set to 0.1. If the command is mandatory and has no default, the program will stop with an error, telling you the input is missing.

- COMMENTS. A single exclamation mark ! in a line causes all the rest of that line to be ignored (and so can be used as a comment). This is true everywhere in the input. Note that (unlike TABLO Input files – see section 4.1.4 of GPD-2) a comment does not need a finishing !. Each comment finishes at the end of the line it starts on (though it can be continued by putting ! at the start of the next line).
  If you actually want an exclamation mark (for example, in the verbal description), just put in a pair !!, which will be treated as one and the rest of the line will still be read. For example, the line
  `Shocks only given to domestic inputs!! Foreign ones given no shock.`
  could be included as part of the verbal description (see section 8.1).

- ORDER: Although the idea is to have order-independent input, the order of the statements can be important in choosing a closure. Then the relevant statements (those beginning 'exogenous', 'endogenous', 'rest' or 'swap') are processed in the order they appear in the command file.

  [Clearly the order can affect the result and/or validity of the commands, as the following example makes clear:

```
exogenous y ;
endogenous y 1 ;
swap y 1 = z 5 ;  ]
```

- VARIABLE COMPONENTS: In the Command file statements, the notation

  <v1 [component_list]> <v2 [component_list]>

  covers three possibilities, as described below.

a) Components can be indicated by number, for example

```
x3  2-12  15-20
```

   See section 5.3 for the meaning of these numbers.

b) The variable name can be followed by sets and/or element names as arguments, for example

```
p_XF("labor", SECT)
```

   In this case there must be no space between the end of the variable name and the "(" starting the arguments.

c) If a variable name is not followed by component numbers as in (a) above or by arguments as in (b) above, this means **all** components of the variable.

- SHOCKS: Full details of the Command file syntax and semantics for "shock" statements in Command files can be found in sections 5.5 to 5.8.

  Where possible, we recommend that you specify components using sets, subset and elements as arguments as in, for example,

shock p_XINTFAC("capital",IND) = … ; ! preferred

rather than using component numbers, as in, for example,

shock p_XINTFAC 2 4 = .. ; ! less easy to use and understand

- <CMF> In writing Command files you can use the shorthand notation **<cmf>** to stand for the command file stem (that is, the CMF name without the suffix .CMF or .cmf). For example

```
equations file = <cmf> ;
updated file iodata = <cmf>abc.upd ;
```

If the Command file is named SJLB.CMF, the equations file is SJLB.EQ4, and updated file `iodata` is SJLBabc.upd See section 2.5.4 for details.

- TAB characters and other control characters (ASCII characters 0-31 and 127) can cause problems in Command files (and Stored-input files). In Release 8.0, TAB characters are replaced by a single space. Most control characters are replaced by a single space but will cause a warning message. The program will stop with an error message if it finds a Control-Z character before the end of the file if there is text after it, or if there are two or more in the file. There is no testing for other characters (ASCII characters 128-255) but foreign language characters which use these ASCII characters would cause problems especially if used in filenames – see section 5.9.2 in GPD-1.

2.8 Eliminating Syntax Errors in GEMPACK Command Files

When you run GEMSIM, a TABLO-generated program, or SAGEM, and take input from a GEMPACK Command file, the whole Command file is checked to see that the syntax of all statements (or commands) in it is as expected (that is, conforms to the syntax laid down in chapter 18). Any syntax errors are pointed out; the message always includes '**%%**' at the start. For example, if a GEMSIM Command file contains the statement "somution file = sjlb ;" (a typing slip), you will see the message

```
%% Unknown keyword 'somution'
```

The program only begins to process (that is, act on) the commands in the Command file if no syntax errors are found.

While this syntax checking is going on, the whole Command file is echoed to the screen, together with any syntax error messages. If you have a large Command file, syntax errors may get lost as the Command file flashes past on the screen. However GEMPACK programs using a Command file always create at least a temporary LOG file (even if you didn't ask for one), and they tell you the name of the LOG file at the end of their run. So, to identify where errors occurred during processing of your Command file, you can search this Log file for '%%' to identify all syntax errors, which you can then correct by editing the Command file. [If you run simulations under WinGEM, it makes it even easier to identify where errors occur.]

Section 3.9.2 of GPD-1 contains hands-on examples showing how you can identify and correct errors in Command files.

CHAPTER 3

3. TABLO Input Files and Auxiliary Files

The implementation of a model in GEMPACK is discussed in chapter 2 of GPD-1 – see section 2.5.1 about TABLO-generated programs and section 2.5.2 about the program GEMSIM.

One of the few respects in which GEMSIM and TABLO-generated programs are different is that:

- For a TABLO-generated program, you have the executable image (.EXE) file [see section 1.2 of GPD-2] and the associated Auxiliary files for a given model.
- For GEMSIM you use the program GEMSIM and the GEMSIM Auxiliary files associated with your model.

In either case, the Auxiliary files incorporate all the relevant details from the TABLO Input file. If you carry out condensation, the names of the Auxiliary files may be different from the name of the TABLO Input file – see section 3.3 below.

3.1 GEMSIM and GEMSIM Auxiliary Files

When you run TABLO with a TABLO Input file and ask for output for GEMSIM (**PGS** option), TABLO writes two GEMSIM Auxiliary files with suffixes '.GSS' and '.GST'. These files contain all the information about the model which was in your TABLO Input file.

For example, if you run TABLO with the TABLO Input file SJ.TAB and ask for GEMSIM output, TABLO will write the Auxiliary files SJ.GSS and SJ.GST.

To run a simulation using the Command file SJLB.CMF for example from the Command prompt, you can enter

gemsim -cmf sjlb.cmf

If you are running GEMSIM from a GEMPACK Command file, you **must** give the name of the Auxiliary files via a Command file statement of the form

Auxiliary files = <file_name> **;** **! mandatory for GEMSIM**

For example,

auxiliary files = sj ;

(The relevant suffixes are added automatically by GEMSIM.)[8]

You can include **directory and path information** in the <file_name>.

For example, suppose you are working in the directory C:\XXX and your Command file SJLB.CMF is also in the same directory, but the GEMSIM Auxiliary files SJ.GSS and SJ.GST are in directory C:\SJ. To run the program for example at the command prompt, you can enter the command:

gemsim -cmf sjlb.cmf

To enable GEMSIM to find the Auxiliary files in directory C:\SJ, you need to include in your Command file the statement:

auxiliary files = c:\sj\sj ;

Note: If you change the name of your TABLO Input file, remember to change the name of the Auxiliary files in the Command file.

[8] If you run GEMSIM interactively or via a Stored-input file, you give the name of the GEMSIM Auxiliary files (the GEMSIM Statement and Table files) after the first menu.

3.2 TABLO-generated Programs and Auxiliary Files

When you run TABLO with a TABLO Input file and ask to write a TABLO-generated program (option **WFP**), TABLO writes the Fortran program (usually with suffix '.FOR') and two Auxiliary files with suffixes '.AXS' and '.AXT'. These files contain all the information about the model which was in your TABLO Input file.

> For example, if you run TABLO with the TABLO Input file SJ.TAB and ask for a TABLO-generated program, TABLO will create SJ.FOR and the Auxiliary files SJ.AXS and SJ.AXT.

When you compile and link the TABLO-generated Fortran program, the TABLO-generated executable-image (for example, SJ.EXE) is created (see Step 1(b) in section 2.5.1 of GPD-1 and section 1.2 of GPD-2).

When you run the executable-image (for example. SJ.EXE) of the TABLO-generated program, the program needs to be able to find its Auxiliary files. How this is done depends on the sort of machine the TABLO-generated program was written on. There are two cases:

- Windows PCs using any version of LF90 or version 5.60 or later of LF95. (See section 3.2.1 below for details.)
- Unix machines or a Windows PC using version 5.50 of LF95. This also applies for TABLO-generated programs built using Release 7 or earlier of GEMPACK. (See section 3.2.2 below for details.)

3.2.1 Windows PCs (Except for Version 5.50 of LF95)

This section applies to Release 8 or later of GEMPACK on Windows PCs using any version of LF90 or version 5.60 or later of LF95.

TABLO-generated programs **always** expect their Auxiliary files to be in the same directory and with the same name. Any "Auxiliary file = … ;" statement in the Command file is ignored.[9]

> For example, suppose that you are running the TABLO-generated program C:\MYMODELS\SJ.EXE . The Auxiliary files must be in the same directory (C:\MYMODELS) and have the same name "SJ". That is, they must be C:\MYMODELS\SJ.AXS and C:\MYMODELS\SJ.AXT .

If you create the EXE in one directory, and later move it to another directory, you **must** move the Auxiliary files to the same directory.

In summary:

1. TABLO-generated programs and their Auxiliary files **must be in the same directory**.
2. There is no need to use any "Auxiliary files = … ;" statement for TABLO-generated programs. The **Auxiliary file statement will be ignored**.

3.2.2 Unix Machines and Windows PCs using Version 5.50 of LF95

This section applies on Unix machines or on Windows PCs using version 5.50 of LF95.

It also applies for TABLO-generated programs built using Release 7 or earlier of GEMPACK.

When you run TABLO, the name of the TABLO-generated program and its Auxiliary files is taken from your response to the prompt headed "PROGRAM FOR THIS MODEL" during the Code stage of TABLO. Unless there is an Auxiliary file statement in the Command file, the TABLO-generated program uses this response to find its Auxiliary files. Usually this is sufficient for the TABLO-generated EXE to find the Auxiliary files.

However, occasionally (perhaps when you are running the program from a file server or another directory), you may need to tell the program where to find these Auxiliary files. The Command file statement is.

Auxiliary files = <file_name> ; **! optional for TABLO-generated programs**

where <file_name> is the name of the Auxiliary files (excluding suffix).[10] You can include directory and path information in <file_name>.

---

[9] We are grateful to Robert McDougall for pointing out the problems associated with finding the Auxiliary files when running a TABLO-generated program on the DOS Path prior to this change.

Examples will make this clear.

Example - PROGRAM FOR THIS MODEL = **SJ**

Suppose that you were in directory C:\MYMODELS when you ran TABLO.

Suppose that you responded "SJ" to the prompt headed "PROGRAM FOR THIS MODEL". Then the Auxiliary files will be SJ.AXS and SJ.AXT and will be in directory C:\MYMODELS.

- If you are in this same directory when you issue the command
  sj -cmf sjlb.cmf
  to run SJ.EXE taking inputs from Command file SJLB.CMF, the TABLO-generated EXE SJ.EXE will expect the Auxiliary files to be SJ.AXS and SJ.AXT in the current directory C:\MYMODELS.
- If you are in directory C:\TEMP which you issue the command
  c:\mymodels\sj -cmf sjlb.cmf
  to run SJ.EXE, your Command file SJLB.CMF must be in your current working directory C:\TEMP. In this case you will need the statement "Auxiliary files = c:\mymodels\sj" in SJLB.CMF in order for the program to find its Auxiliary files.

Example - PROGRAM FOR THIS MODEL = **C:\MYMODELS\SJ**

Suppose that you were in directory C:\MYMODELS when you ran TABLO.

Suppose that you responded "C:\MYMODELS\SJ" to the prompt headed "PROGRAM FOR THIS MODEL" when you ran TABLO. Again the Auxiliary files will be called SJ.AXS and SJ.AXT and will be in directory C:\MYMODELS. Because the full path name of the Auxiliary files is contained in this response, the TABLO-generated EXE SJ.EXE will always find its Auxiliary files (provided you do not move them from directory C:\MYMODELS). Even if you are in a different working directory when you run C:\MYMODELS\SJ.EXE, the program will find the Auxiliary files.

If you create the EXE in one directory, and later move it to another directory, we suggest that you also move the Auxiliary files to the same directory. You also need to put an "Auxiliary files = ...;" statement in your Command file.

Note: If you change the name of your TABLO Input file, remember to change the name of the Auxiliary files in the Command file.

In summary:

1. If there is an "Auxiliary files = … ;" statement in the Command file, the program uses this to find the Auxiliary files.
2. If not, the program uses the response to the prompt headed "PROGRAM FOR THIS MODEL" when TABLO was run to find the Auxiliary files.

3.3 How are the Names of the Auxiliary Files Determined?

Modellers are sometimes confused about the names of the Auxiliary files. Usually they are the same as the name of the TABLO Input file (though with different suffixes). But sometimes they are different. In all cases,

the names of the Auxiliary files are determined when you run TABLO.

[And this applies equally well if your TABLO Input file is for an economic model or for data manipulation.]

- If you ran TABLO in the usual simple way, you will start from TABLO Input file XX.TAB and produce either XX.FOR, XX.AXS, XX.AXT and XX.EXE (if TABLO produced a TABLO-generated program) or XX.GSS and XX.GST (if TABLO produced output for GEMSIM).
- If you carried out condensation starting from file XX.TAB, you may use a name different from XX for the output files from TABLO (either the TABLO-generated program or the GEMSIM Auxiliary files). For example, you may produce GEMSIM Auxiliary files called XXCON.GSS and .GST to indicate that they come from a condensation of XX.TAB.

---

[10] Alternatively you can use the option NAX in the initial options menu of TABLO-generated programs
NAX Name AXS,T files

In all cases, the names of the Auxiliary files are determined by your response to the last prompt during the Code stage of TABLO. If you are producing a TABLO-generated program, this prompt is headed "PROGRAM FOR THIS MODEL". If you are producing output for GEMSIM, this prompt is headed "NAME OF GEMSIM STATEMENT FILE".

Note that, often, you enter a Carriage-Return in response to this prompt to indicate that you wish to accept the default offered. Then the names of the Auxiliary files are determined as if you had given the default response offered.[11]

3.4 Check that Auxiliary Files are Correct

TABLO-generated programs check that the Fortran file from which the executable image was made (see section 1.2 of GPD-2) and the Auxiliary files (AXS and AXT files) were created at the same time. This is to avoid inadvertently accessing AXS,AXT files which do not really belong to the TABLO-generated program you are running.

For example if you run TABLO but forget to compile and link the TABLO-generated program, you will probably get an error message like the one in the box below. This is because the old executable image of the TABLO-generated program does not match the new Auxiliary files.[12]

```
%% Program and Auxiliary files were created at different times.
Program was created at 08:46:20 on 05-SEP-2002 while
AXS,AXT files were created at 11:51:53 on 05-SEP-2002.

 (ERROR RETURN FROM ROUTINE: TGCAXD)
 (E-PG date problem)
 (ERROR RETURN FROM ROUTINE: Main Program)
```

GEMSIM checks that the GEMSIM Auxiliary files (GSS,GST files) you are accessing were both created at the same time.

3.4.1 Model Information (MIN) File

When TABLO produces a TABLO-generated program, it produces a file called the Model Information file with suffix (.MIN). This file is a text file used by the programs RunGEM, AnalyseGE and RunDynam – see sections 2.5, 2.6 and 2.7 in GPD-4.[13]

3.4.2 Compatibility of Various Files

As described above, dates put on the .FOR, .AXS and .AXT files have been used for several versions of GEMPACK to detect when these were not created at the same time. The same date is put on the .MIN file. Programs such as RunGEM and AnalyseGE use this to detect if the .MIN file was not created at the same time as the AXT (or other) files. Similarly for the .GSS, .GST and .MIN files produced when TABLO produces output for GEMSIM.

3.5 Carrying out Simulations on Other Machines

Once you have built a model, you may be able to move the software for carrying out simulations with it to other machines **with the same operating system** but on which GEMPACK is not installed. For example,

---

[11] In most cases, the default offered comes from the name of the Information file chosen during the Check stage of TABLO. If you are doing condensation and choose a name different from the TAB file for the Auxiliary files (see, for example, the XXCON example in the text), the new name (eg, XXCON) is usually given as the name of the Information file.

[12] A simple visual check you can carry out is that the time on the EXE file should be later than the time on the AXS, AXT files because LTG is run after the TABLO run to create the FOR, AXS, AXT, MIN files.

[13] The Model Information file was introduced in Release 6.0.

- you may wish to allow colleagues outside your organisation to carry out simulations with your model.
- you may wish to allow students to carry out simulations with your model in a computer laboratory, possibly via RunGEM (see section 2.5 of GPD-4).

If you solve your model using a TABLO-generated program, it is relatively easy to move your TABLO-generated program. The machine to which you copy the program does not need to have GEMPACK or Fortran installed, and a GEMPACK licence may not be required. We explain this in more detail below.

However if you solve your model using GEMSIM, it is usually not practical to copy the GEMSIM Auxiliary files to another machine in order to carry out simulations there. If the other machine has GEMPACK installed, it is simpler to run TABLO and then GEMSIM there. If the other machine does not have GEMPACK installed, you cannot use the GEMSIM Auxiliary files since GEMSIM is also needed. If you have in mind a colleague who does not have a GEMPACK licence, you are not allowed to give that colleague a copy of GEMSIM since that would breach the terms of your GEMPACK licence. So below we concentrate on copying TABLO-generated programs.

3.5.1 Copying TABLO-generated Programs to Other Computers (Same Operating System)

To transfer a TABLO-generated program, you must copy

1. the program executable-image eg SJ.EXE on PCs, (or sj for Unix)
2. the Auxiliary Statement and Table files (these have suffixes '.AXS' and '.AXT' respectively) to the other machine since the TABLO-generated program will not run without them, eg SJ.AXS and SJ.AXT
3. the Model Information file (this has suffix '.MIN') if you want to use RunGEM or AnalyseGE. eg SJ.MIN

Of course you will also need to copy any data files used in the simulation, and any relevant Command files or Stored-input files.

You can also copy the executable-image of GEMPIE to the other machine, or install ViewSOL on the other machine, in order to see the results of your simulation.

Note that, because the program executable-image, the Auxiliary Statement and Table files are binary files, you cannot move TABLO-generated programs produced on one type of machine (for example, a Unix machine) directly to another type of machine (for example, a PC). To move a model from one type of machine to another on which GEMPACK is installed, follow the procedure described in section 11.1 of GPD-4 (which involves transferring the TABLO Input file and the data files).

On a Windows PC, if the relevant Lahey Fortran is not installed on the new machine and on the path, you will need "stand-alone" EXE files for the TABLO-generated program.

- For Release 7.0 or later, executable images compiled with LF90 or LF95 are automatically stand-alone.
- However, Release 6.0 (or earlier) executable images and Release 7.0 executable images compiled with F77L3 are not automatically stand-alone. For such TABLO-generated programs, use LTGS instead of LTG (see the Release 6 or Release 7 versions of GPD-6).

For example, if you are using RunGEM and a TABLO-generated program of some model for your students to use on a network, you will need to make sure that the EXE file is a stand-alone one.

3.5.2 GEMPACK Licence May Be Required

TABLO and GEMSIM always require a GEMPACK licence.

A TABLO-generated program may require a GEMPACK licence if the model is too large – see section 1.9.5 of GPD-1.

Windows programs such as ViewSOL and AnalyseGE may require a GEMPACK licence if the Solution file is large (see the relevant Help file).

Other GEMPACK utility programs (including GEMPIE, SEEHAR and MODHAR) do not require a GEMPACK licence.

CHAPTER 4

4. Data Files, Updated Data Files and Display Files

TABLO-generated programs and GEMSIM can read data from data files, and can write data to data files or Display files. Section 4.1 describes Command file statements for data files.

When TABLO-generated programs and GEMSIM carry out a simulation, they normally produce updated data files (which are post-simulation versions of the corresponding original data files). Section 4.2 describes Command file statements for updated data files.

Display file options are in section 4.3 while section 4.4 discusses how TABLO-generated programs and GEMSIM check data read from data files and gives the Command file syntax related to "Check-on-read".

4.1 Data Files and Logical Files

In TABLO Input files, FILE statements specify the **logical** file names of data files. In a Command file to run a simulation these logical file names must be linked to an **actual** file name of a file on your computer.[14] The Command file statement is

```
file <logical_name> = <actual_name> ;
```

The purpose of a "file" statement in a Command file is to **make the connection between the logical file name (as in the TABLO Input file) and an actual file on your computer**.

Whenever you have a "read" statement in a TABLO Input file, it refers to the logical file name (for example "read DVHOUS from file iodata … ;"). When GEMSIM or the TABLO-generated program runs it will read this data from the actual file which is linked to the logical file name via the "file" statement in the Command file.

Similarly for "write" statements.

Example 1 – ORANIG

In the ORANIG98 model, there is one input data file with logical file name MDATA. This is defined in the TABLO Input file ORANIG98.TAB (see section 1.4.2 of GPD-8) via the statement

```
File MDATA  # Data file # ;
```

In the TABLO Input file you will see a statement

```
Read V4BAS from file MDATA Header "4BAS" ;
```

[V4BAS(c) is the dollar value of exports of commodity c at basic values.]

The actual data read into Coefficient V4BAS depends on the "file MDATA = … " statement in the Command file. You may have several different data files for ORANIG, including one called OG98.DAT with 1998 data and another called OG97.DAT with 1997 data. If you want to base a simulation on the 1998 data, you will include the statement

```
File MDATA = og98.dat ;
```

while you will include the statement

```
File MDATA = og97.dat ;
```

if you wish to start your simulation from the 1997 data.

[14] It is possible to put the actual file name in your TABLO Input file (see section 3.5 of GPD-2) and in this case a file statement is not necessary in the Command file. However this is very inflexible since it means that in order to change your data file at simulation run-time you would need to re-run TABLO before running the simulation.

This illustrates the main reason for only having logical file names in the TABLO Input file (rather than actual file names). This makes the TABLO Input files more flexible since the corresponding TABLO-generated programs can then be used with several different actual files without needing to be changed.

Example 2 – Stylized Johansen

In the TABLO Input file for Stylized Johansen (see section 3.3.3 of GPD-1) there is just one FILE statement:

```
FILE iodata  # input-output data for the model # ;
```

The logical filename is *iodata* but the file on your computer has actual name SJ.DAT. So in your Command file you can put a statement:

```
file  iodata  = sj.dat ;
```

In another simulation with the same model, you may start from the data updated after an earlier simulation. Then you would include a statement

```
file iodata = sjlb.upd ;
```

to indicate that this time the data which the TABLO Input file says is to be read from logical file "iodata" will be read from the actual file called sjlb.upd.

Example 3 – GTAP

In the TABLO Input file GTAP61.TAB supplied with the GEMPACK examples (see section 1.6.1 of GPD-8), there are three FILE statements. The logical names are GTAPDATA (holds the input-output data for each region and the bilateral trade data), GTAPSETS (holds the names of the elements of the various SETs) and GTAPPARM (holds the values of the parameters). There are many different data sets for GTAP, representing different aggregations (that is, sets of regions and commodities) and also representing different years. In a Command file for a simulation with this model, there are three ***file*** statements. For example, in GEX15.CMF (see section 7.1.1), the statements are

```
! Input data files
file GTAPDATA = gdatc3x3.har ;
file GTAPSETS = gsetc3x3.har ;
file GTAPPARM = gparc3x3.har ;
```

For each logical file name declared in the TABLO Input file, you need the corresponding ***file*** statement in your Command file. A ***file*** statement is required for each original data file and for each new file.

4.1.1 Input and Output Data files

TABLO-generated programs and GEMSIM can read data from files (input files) and can write data to other files (output files). Input and output data files are distinguished in the TABLO Input file when the corresponding logical file is declared (see section 3.5 of GPD-2). Output files (those which are written to) are declared with the **(NEW)** qualifier as in (see, for example, ORANIG98.TAB in section 1.4.2 of GPD-8)

```
File (NEW) SUMMARY # Output file for checking data # ;  !TABLO Input file !
```

In the Command file, there is a corresponding statement, for example,

```
File SUMMARY = sum1987.dat ;       !Command file
```

In Command files, the syntax of the ***file*** statement connecting a logical file (from the TABLO Input file) to the actual name is the same whether the file is intended for input or output. That is, when you see the statement

```
File  FILE3 = f3.dat ;        ! Command file
```

in a Command file, you must look at the TABLO Input file to know whether the file f3.dat is being read from (that is, is an input file) or the file f3.dat is being written to (that is, is an output file).

If FILE3 corresponds to an input file, the file f3.dat must exist before the TABLO-generated or GEMSIM program runs.

If FILE3 corresponds to an output file, the file f3.dat will be created by the TABLO-generated program or GEMSIM (and any pre-existing version of that file will be deleted). In a ***file*** statement for a new file in a Command file, you choose the name of the actual file which will be written.

TABLO-generated programs and GEMSIM can also write output to Display files – see section 4.3 below for details.

Input and output files can be Header Array files or text files. When data is written to an output text file, the file is written as a GEMPACK text data file (see chapter 6 of GPD-4).

When TABLO-generated programs and GEMSIM read data from a Header Array file, they can check that the set and element information is as expected – see section 4.4 below for details.

Example - GTAPVIEW

In the TABLO Input file GTPVEW61.TAB (see section 6.8.1) supplied with the GEMPACK examples, there are three input files and two new output files.

Excerpt from TABLO Input file GTPVEW61.TAB

```
File GTAPDATA # file containing all base data #;
File GTAPSETS # file with set specification #;
File GTAPPARM # file containing behavioral parameters #;
File (NEW) GTAPVIEW # file with headers for viewing #;
File (NEW) TAXRATES # file with headers for viewing initial tax rates #;
```

The logical names for the input files are the same as for simulations with GTAP61 (see Example 3 above). The logical names of the two new Header Array files produced are GTAPVIEW (contains summaries, totals and shares) and TAXRATES (reports the *ad valorem* tax rates implied by the data). In a Command file for a running this TAB file, there are three FILE statements for the input data files and two FILE statements for the output files. For example, in GTPV3X3.CMF (see section 6.8.1), the statements are as in the box below.

Excerpt from Command file GTPV3X3.CMF

```
! Input data files
file GTAPDATA = gdatc3x3.har ;
file GTAPSETS = gsetc3x3.har ;
file GTAPPARM = gparc3x3.har ;

! Output files
file GTAPVIEW = <cmf>.har ;
file TAXRATES = <cmf>t.har ;
```

Note the two uses of **<cmf>** (see section 2.5.4 above) for the output files. The new GTAPVIEW file gets the same name as the Command file (that is, GTPV3X3.HAR when GTPV3X3.CMF is run) and the new TAXRATES file gets that name with a "t" (for "tax") at the end (that is, GTPV3X3T.HAR when GTPV3X3.CMF is run). Adding the "t" works well to keep the output files from checking different data sets all distinct as long as you don't have a second Command file whose basic name has a "t" added. [For example, GTPV3X3.CMF and GTPV3X3T.CMF would cause some overwriting of output files.] Subject to that proviso, the two lines in the Command file for output files GTAPVIEW and TAXRATES do not need to be changed when you run GTAPVIEW on a different group of GTAP data sets (for example, a different aggregation).[15]

Alternatively you might like to use the statements

```
! Output filenames in Command file
file GTAPVIEW = <cmf>-view.har ;
file TAXRATES = <cmf>-tax.har ;
```

to specify the names of the output files. These would produce output files GTPV3X3-VIEW.HAR and GTPV3X3-TAX.HAR. GEMPACK software can handle such long file names on most machines (see section 5.9 of GPD-1).

[15] You are not allowed to specify the same name (for example, <cmf>.har) for both the GTAPVIEW and TAXRATES output file. If you do, the program will stop with an error. [However, it is possible to link the same input Header Array file to two different logical files, though this is only done rarely.]

4.2 Updated Data Files

If you have a FILE statement in your TABLO Input file, and

- if the values of one or more coefficients are read from that logical file are updated using an UPDATE statement, or
- the values of some Levels variable are read from that logical file (in which case there is an automatic UPDATE statement – see section 2.2.2 of GPD-2),

then you need a corresponding Command file statement:

updated file <logical_name> = <actual_name> ;

If all the coefficients on the logical file are parameters (that is, are not updated), then you do not need this type of statement.

The **logical name** is the name in your TABLO Input file. The **actual name** is the name of the new updated version of the original data file which will be created on your computer disk after the simulation. This updated data file contains the original data updated using the percentage changes or ordinary changes in your solution variables. Only TABLO-generated programs and GEMSIM can update files, not SAGEM.

You can think of the updated data as representing the state of the economy as it would be once the simulation shocks have worked their way through the economy (see, for example, section 2.9 of GPD-1).

For example, the statement

```
updated file iodata = sjlb.upd ;
```

ensures that the updated version of the data on input file with logical name `iodata` will be written to the file SJLB.UPD.

4.2.1 Naming Updated Files

It is good practice to choose names for the updated data files which are similar to, or identical to, the name of the Command file. For example, in the Command file SJLB.CMF for a simulation with the Stylized Johansen model the updated version of *iodata* is called SJLB.UPD.

An alternative way of writing the updated file statement is to use the **<cmf>** abbreviation – see section 2.5.4

```
updated file iodata = <cmf>.upd ;
```

We strongly suggest that you always choose names with suffixes of exactly the same length as the other standard GEMPACK file suffixes on your machine. On most machines (including Windows PCs and Unix machines), this means choosing a suffix of length 3 (following the '.'); we find the suffix **.upd** ideal.[16]

On machines where long file names are allowed, it is practicable to include an indication of the various features of the simulation that affect the updated data (the closure, shocks, possibly the original data base if there are several you could have used, and the method and steps). For example, with Stylized Johansen, the updated data might be called

`sjlb10_124e.upd`

to indicate that labor has been increased by 10 percent ("lb10") and that the simulation used 1,2,4-step Euler calculations ("124e").

Example – GTAP

In the TABLO Input file GTAP61.TAB supplied with the GEMPACK examples (see section 1.6.1 of GPD-8), there are three FILE statements. The logical names are GTAPDATA (holds the input-output for each region and the bilateral trade data), GTAPSETS (holds the names of the elements of the various SETs) and GTAPPARM (holds the values of the parameters). Of these, only the data in the

[16] Choosing a suffix of the same length as other suffixes ensures that the program can provide default names for intermediate updated data files (see section 4.5.1 below).

GTAPDATA file is updated. [None of the data on the GTAPSETS or GTAPPARM files is updated.] Hence, in a Command file for a simulation with this model there only needs to be one "updated file" statement. For example, in GEX15.CMF (see section 7.1.1), the statement is

```
! Updated data file
updated file GTAPDATA = <cmf>.upd ;
```

Note the use of <cmf> (see section 2.5.4 above) in this statement. When GEX15.CMF is run, the updated GTAPDATA file produced is called GEX15.UPD. This statement does not need to be changed in the related Command files (for example, GEX15I.CMF – see section 7.1.1). When GEX15I.CMF is run, the updated GTAPDATA file produced is called GEX15I.UPD.

4.2.2 Updated Data Read from the Terminal

A file holding updated values of data read from the terminal is produced in a multi-step simulation whenever data is read from the terminal. So you should include a statement

```
updated terminal data = <file_name> ;
```

in your Command file precisely in this case. The file produced will be a text data file (see chapter 6 of GPD-4).

4.2.3 Intermediate Data Files

In rare circumstances, GEMSIM or a TABLO-generated program may need to know the name of an intermediate file to be used in the course of a simulation. If you are working from a Command file, the program will usually generate and use a default name as described in section 4.5.1. Thus you probably do not need to know about this. In the unlikely event that you do, details can be found in section 4.5 below.

4.2.4 Long Names on Updated Data Files

If the original data file is a Header Array file, the updated version is also a Header Array file. When GEMSIM or a TABLO-generated program writes the updated data file, it transfers the long name (see section 3.1.2 of GPD-4) from the relevant header on the original file to the updated data file. If there is no long name (that is, if the long name is blank), the program will use the Coefficient Label (the text between # # when the Coefficient is declared in the TABLO Input file) as the long name on the updated data file.[17]

4.3 Display Files

If you want to see the values of a Coefficient in your TABLO Input file, you can write them to an output file (for example, "write DVHOUS to file out1 ;") or you can ask to have them displayed via a display statement as in "**`display  DVHOUS ;"`**

The output from all display statements is written to a single new output file called the **Display file**. The name of the display file can be given in the Command file as

```
display file = <file_name> ;      ! Include the suffix.
```

If this statement is omitted, a default name is used for the display file – this is the command file stem plus the suffix ".dis" - provided the Command file ends with .CMF or .cmf (see section 2.5.3). For example, if the Command file is sjlb.cmf, then the display file is sjlb.dis.

Matrices and higher dimensional arrays are written in a Display file in a tabular form which is designed to be easily readable when the Display file is printed (or viewed on the screen in a text editor using a fixed-width font). This is different from the output which results from a "write" statement: then the values are either written as a header on a Header Array file or as a GEMPACK text data file.

A Display file cannot be read again by GEMPACK programs. This contrasts with the files written via "write" statements, the text or Header Array versions of which can be read by GEMPACK programs.

---

[17] This was introduced in Release 8.0.

Real data is written via a Display statement in exactly the same form as labelled output from SEEHAR (see section 4.1 of GPD-4). In particular, see section 4.1.2 of GPD-4 if you wish to import DISPLAY data into a spreadsheet.

4.3.1 Options for Display Files

The normal format for any DISPLAYs is to show real numbers with 6 decimal places and to put at most 58 lines per page and at most 132 or 80 (depending on the machine – PCs usually use 80) characters per line. There are Command file statements which allow you to vary these.[18]

Page width can be varied between 70 and 200 characters and page length must be at least 20 lines. The number of figures after the decimal point can be between 0 and 10. The related Command file statements are

display width = <integer> ; ! Example. Display width = 130 ;

display length = <integer> ; ! Example. Display length = 50 ;

display decimals = <integer> ; ! Example. Display decimals = 3 ;

Normally in a display file, new pages will be started only when the current page is close to the end. If you put the statement

"**DPN = no ;**"

in your Command file, each new coefficient displayed starts on a new page.[19]

Normally in a display file, one-dimensional arrays are displayed as a row (that is, across the page). If you put the statement

"**D1C = yes ;**"

in your Command file, such arrays will be displayed as a column (that is, down the page).

Normally in a display file, if one row contains the same values as the previous one, a message indicating this is shown rather than the actual values being repeated. This can apply to several rows in succession. If you put the statement

"**DOI = yes ;**"

in your Command file, no "identical" row messages will be shown: instead all values will be shown.

4.4 Checking Set and Element Information when Reading Data

When GEMSIM and TABLO-generated programs read data, they know which coefficient they are reading data for, how many arguments that coefficient has, which sets these arguments range over and, possibly, the names of the elements of these sets. If the data is being read from a Header Array file, and if the data at the relevant header also has set and element information on the Header Array file (that is, if the array is of type **RE** – see section 5.1 of GPD-4), these programs can check whether this set/element/coefficient information as held on the file agrees with what they expect.

For example, consider the following part of a TABLO Input file:

```
Set SECT (agriculture, manufactures, services) ;
Coefficient (All,c,SECT) DVHOUS(c) ;
File iodata ;
Read DVHOUS from file iodata Header "DVHS" ;
```

If the data at header DVHS on the relevant Header Array file is of type RE, the file may contain some or all of:

---

[18] There are also interactive options corresponding to the Command file statements described in this section – see section 14.5 for details.

[19] The default was changed for Release 7.0. Previously "dpn = no ;" was the default but, from Release 7.0, "dpn = yes ;" is the default. The peculiar abbreviations DPN, D1C and DOI (see later in this section) come from the interactive version of these options – see section 14.5 for details.

- the name of the coefficient usually associated with the data,
- the number of arguments of that coefficient,
- the names of the sets over which these arguments range, and
- the names of the elements of these sets.

GEMSIM and TABLO-generated programs always check that the amount of data at the header is as expected (see section 4.10.1 of GPD-2). Since SECT has size 3, the READ from header DVHS will only go ahead if there are just 3 numbers at the header (that is, provided the data at that header is a 1-dimensional array of size 3, or if it has some other arguments of size 1 – for example, if it is of size 3x1 or 1x3 etc). As indicated in section 4.10.1 of GPD-2, the first number on the file at header "DVHS" will be assigned to DVHOUS("agriculture") since agriculture is the first element in the set SECT, as defined in the TABLO Input file. Similarly, the second and third numbers on the file at that header will be assigned to DVHOUS("manufactures") and DVHOUS("services") respectively.

But suppose, for example, that the RE information on the file was that the coefficient stored there has one argument ranging over a set called SECT whose elements were *agriculture, services and manufactures* (in that order – note that this is different from the order in the TABLO Input file), there could be a serious problem since the second number on the file will be associated by GEMSIM or the TABLO-generated program with what it thinks is the second sector (namely *manufactures*[20]) but (since the second sector on the file is *services*), presumably this second number really refers to *services* and not *manufactures*.

The software can check[21] whether all of the four things above (coefficient name, number of arguments, names of sets, names of elements) agree with those on the file.

We realise that there may be many circumstances when you do not want such checking to occur. For example,

1. a data manipulation TABLO Input file may be deliberately changing the set and element information,
2. sometimes several coefficients (with different names) are read from the same header,
3. some models are set up to have commodity and industry labels in different languages so the same data file may be used with, on the one hand, English names for the commodities and industries and, on the other hand, Vietnamese names for these.

In these and other cases you may prefer to turn off some or all of the above checking. For example, in case 2 above you may want to turn off checking of coefficient names but leave other checking on. In case 3 above, you may want to turn off checking of element names but leave other checking on,

For this reason we have introduced Command file statements which make it easy for you to specify the amount of this checking. The relevant statements are shown below.

```
Check-on-read coefficients  =  yes|warn|no ;
Check-on-read sets          =  yes|warn|no ;
Check-on-read elements      =  yes|warn|no ;
```

These say how checking of coefficient names, set names and element names should be done.

- If **no** is put after the "=", no checking is done.
- If **warn** is put after the "=", only a warning is issued (and the program continues) if there is a mismatch between the information expected and that found on the file.
- If **yes** is put after the "=", a fatal error occurs (and the program stops) if there is a mismatch between the information expected and that on the file.

There is a way of specifying all three at once if they are all the same level. The statement

```
Check-on-read all  =  yes|warn|no ;
```

sets all three (coefficients, sets, elements) at the same level of checking. For example *check-on-read all = yes ;* means that a fatal error will occur if there is any mismatch in coefficient names, set names or element names.

---

[20] This is because of the order of the elements in the declaration of the set SECT in the TABLO Input file.

[21] This was a new feature for Release 6.0 of GEMPACK.

Because reads go ahead without error if there are extra dimensions of size 1 involved (for example, the read into DVHOUS in the example at the start of this section will go ahead if on the file there is a 1-dimensional array of size 3, or a 2-dimensional array of size 1x3 or 3x1), there is also the statement

```
Check-on-read exact  =  yes|no ;
```

Here **yes** requires exact matches between the size of the array expected and found so would cause an error or warning (depending on the other settings[22]) if there are dimensions of size 1 different between that expected and that found. [For example, "check-on-read exact = yes ;" would cause an error or warning in the DVHOUS example if the data on the file is of size 3x1.]

On the other hand, **`check-on-read exact = no ;`** allows extra dimensions of size 1. Indeed, with this setting, single element names are not checked either. For example, with set SECT as in the DVHOUS example given earlier, consider also the statements

```
Coefficient (All,c,SECT)(all,i,SECT) DVIND(c,i) ;
Read (All,c,SECT) DVIND(c,"services") from file
      iodata Header "DVIN" ;
```

If the data at header DVIN is an array of size three and the set/element information there is for a 1-dimensional array ranging over the set SECT with elements as in the DVHOUS example,

- if **`check-on-read exact = yes ;`** has been specified, this would result in a fatal error or warning, but
- if **`check-on-read exact = no ;`** has been specified, no error or warning would occur (nor would any error or warning occur even if the set/element information on the file was for a 2-dimensional coefficient DVIND(SECT,"agriculture") since no checking of element names in the arguments is done when "check-on-read exact = no ;" is specified).

Checking the number of arguments is only carried out if "check-on-read exact =yes ;" is specified in your Command file.

The default settings are:

```
check-on-read sets = warn ;
check-on-read elements = yes ;
check-on-read coefficients = no ;
check-on-read exact = no ;
```

These are what you get if you don't put any check-on-read statements in your Command file. If you find that these defaults are causing fatal errors when no data association errors are in fact occurring, you can change to just warnings (or no checking) for some or all of the three possibilities (coefficients, sets, elements). If you do not wish to do any of this checking, simply put

check-on-read all = no ;

in your Command file.

Note that checking of set elements is only done if the sets in question are the same. Thus, with the above defaults, a fatal error is only generated if one of the sets on the file agrees with that expected but the elements of this set on the file are different from those expected.

There are also the options **CRN**, **CRW**, **CRF** available in the options menu when GEMSIM and TABLO-generated programs run.[23] These are equivalent to putting "check-on-read all = no ;", "check-on-read all = warn ;" and "check-on-read all = yes ;" respectively in your Command file. Each one also implies "check-on-read exact = no ;".

---

[22] If "check-on-read exact = yes ;" has been specified and there is not an exact match between the size of the array expected and found, there may be a set mismatch, an element mismatch, or a set found when an element was expected, or an element found when a set was expected. If an element is involved in the mismatch, this generates an error if "check-on-read elements = yes ;" has been specified. If a set is involved in the mismatch, this generates an error if "check-on-read sets = yes ;" has been specified. Otherwise this generates a warning.

[23] The only option visible in the options menu is **CR**. If you select this, you will see that these options CRN, CRW and CRF can be selected even though they are not shown in the menu.

Note that

- **`check-on-read all = … ;`** does not affect the setting of "check-on-read exact = … ;".
- there is no checking of coefficient names, set names or element names when GEMSIM or a TABLO-generated program reads data from text files (or the terminal).
- these checks are only applied by GEMSIM and TABLO-generated programs; they are not relevant when other GEMPACK programs are running.

4.4.1 Check-on-Read Problems with RunGEM, RunDynam, RunMONASH and RunGTAP

Windows programs such as RunGEM, RunDynam and RunMONASH (see chapter 2 of GPD-4) write Command files based on information provided on various pages of their notebooks. If you wish to specify how check-on-read is done, you can include the relevant statement (for example, "check-on-read all = no ;") in a file called **CMFSTART** (no suffix) in the relevant directory. To find out about this, click on

File | Edit CMFSTART file

in the main menu and then click on the Help button.

RunGTAP has separate CMFSTART files in each version subdirectory. [See the RunGTAP Help file for more details.]

4.4.2 Checks When Reading Set Element Names from a File

The names of set elements can be read in at run time from a Header Array file – see section 4.6.9 of GPD-2 for details. Checks are carried out at run time to see if the element name is a valid name using two different reading styles.

1. **TABLO** style element names are as specified in section 4.2 of GPD-2. These element names must be at most 12 characters long and must start with a letter. Characters in names are allowed to be letters, digits, underscore "_" or character "@". They cannot contain spaces or other characters.
2. Alternatively there are **flexible** style element names containing other characters (apart from spaces).

There is a Command file statement to choose between these two styles[24]:

```
set elements read style = TABLO | flexible ;
```

The default is TABLO style and we encourage you to use this stricter style of element names. However if you have a data file containing set elements containing other characters, you can add the statement

```
set elements read style = flexible ;
```

to your Command file to turn off this checking.

4.4.3 Checks on Run-time Elements used in TABLO Input File

Prior to Release 8, element names occurring in TABLO Input files had to be from sets with fixed elements (that is, sets whose elements were specified in the TABLO Input file). This restriction is removed in Release 8, which allows element names to be from sets with run-time elements (that is, from sets whose elements are read at run-time). See section 4.6.1 of GPD-2 for details about fixed and run-time elements.

Example

```
File Setinfo ;
Set COM # Commodities #   Read Elements from File Setinfo header "COM" ;
Coefficient (all,c,COM)  COEF1(c) ;
Formula COEF1("food") = 23 ;
```

The Formula above would have resulted in a semantic error prior to Release 8.

[24] This statement is new for Release 7.0 of GEMPACK.

In Release 8 or later, TABLO creates a new set called **S@food** containing the single element "**food**" and TABLO adds information to indicate that this set is a subset of the set COM.[25]

Whether or not "food" is an element of COM can only be checked when GEMSIM or the TABLO-generated program runs to carry out the statements in this TABLO Input file. The program (GEMSIM or the TABLO-generated program) checks whether or not "food" is an element of COM when it checks the SUBSET statement (S@food IS SUBSET OF COM) introduced by TABLO. If "food" is not in COM, the error message will say that S@food is not a subset of COM since "food" is not in COM.

See section 4.2.4 of GPD-2 for more details.

4.5 Names of Intermediate Data Files

In rare circumstances, GEMSIM or a TABLO-generated program may need to know the name of an intermediate file to be used in the course of a simulation. If you are working from a Command file, the program will usually generate and use a default name as described in section 4.5.1. Thus you probably do not need to know about this. In the unlikely event that you do, details can be found below.

When carrying out a multi-step calculation, GEMSIM and TABLO-generated programs sometimes need to write intermediate versions of the updated data. For example this could occur possibly after each subinterval (when the simulation is split into several subintervals). Whenever you are extrapolating, intermediate files hold the data as updated after each separate multi-step calculation. Whenever data is read initially from a text file or the terminal, or there are FORMULA(INITIAL)s, a so-called intermediate extra data file may be required to hold updated values between steps or subintervals.[26]

If you are using a GEMPACK Command file, the names of these intermediate files can often be inferred as a default from the names of other files, as explained in section 4.5.1 below

If you are running interactively, these programs will prompt you if and when these files are required. If you are taking inputs from a Stored-input file, you need to anticipate if a prompt for the name of one of these files will be given.

When an Intermediate Updated File is Required

An intermediate updated file is required for any existing Header Array file from which data is read if

- you are extrapolating, and/or
- there are at least 2 subintervals.[27]

An intermediate updated file is never needed for a text file (however then an intermediate extra file will be required, as explained below).

When you use a GEMPACK Command file, you can always include a statement

```
intermediate file <logical_name> = <file_name> ;
```

---

[25] This is as if the following statements were added to the TABLO Input file.

```
SET    S@food (food) ;
SUBSET  S@food   IS SUBSET OF  COM ;
```

[26] The suffixes used on most machines for these files are

- '.ud3', '.ud4' for intermediate data updated after each step,
- '.ud5', '.ud6', '.ud7' for data updated after each separate multi-step calculation when you are extrapolating, and
- '.ud8', '.ud9' for updated data after each subinterval if several are in force.

[27] Prior to Release 7, an intermediate file was also required if option UCS had been selected from the Code Options Menu in TABLO (see section 5.1.3 of GPD-2) when the program was written and there is more than one pass in one or more of the multi-step simulations.

even if you are not sure if it will be needed: if not, GEMSIM or your TABLO-generated code will just ignore it. However, if you follow the advice in section 4.2.1 above about suffixes, you should never need one of these statements since the default chosen by the GEMSIM or your TABLO-generated program should suffice.

When an Intermediate Extra File is Required

An intermediate extra file will be required if you are reading data from a text file or the terminal or if there is a FORMULA(INITIAL); otherwise one will not be needed.

When you use a GEMPACK Command file, you can always include a statement

```
intermediate extra data = <file_name> ;
```

even if you are not sure if it will be needed; if not, GEMSIM or your TABLO-generated program will just ignore it. If data is not read from the terminal or a text file, GEMSIM or TABLO-generated programs will not have a default name to use so you may like to include an "intermediate extra data" statement in this case.

Note that several intermediate updated files may be needed (possibly one for each different Header Array file from which data is read) but only ever one intermediate extra file is required.

4.5.1 Default Names for Intermediate Files When Using a Command File

If you are running a simulation via a GEMPACK Command file, GEMSIM and TABLO-generated programs will usually provide default names for intermediate updated files and the intermediate extra data file when they are needed and you didn't provide one.[28] This avoids the program crashing just because you have not specified a name for one of these files.

These defaults are

GPXXX	Intermediate extra data file
GPXXd	Intermediate updated file (where 'd' is replaced by the number of the file amongst all the FILE statements, for example, GPXX2 for the second one or GPX12 for the twelfth).

The only time when it might not be safe to rely on these default names is if you have two or more programs running at the same time and in the same directory. Then they may overwrite each other's files and the results may be unpredictable unless you specify different names for these files for the different runs in progress at the same time.

[28] These default names will be provided if you use suffixes of the standard length (usually 3) for the updated data files, as recommended in section 4.2.1.

CHAPTER 5

5. Choosing the Closure and Shocks

In this chapter we tell you in more detail about choosing the closure for a simulation (sections 5.1 and 5.2), choosing other sets of variables such as the variables to shock or the endogenous variables to print (section 5.4) and specifying the shocks for a simulation (sections 5.5 to 5.8). Section 5.3 tells you about component numbers for variables with two or more arguments, section 5.9 is about checking the closure and shocks only while section 5.10 is about using Levels variables names in a Command file.

This entire chapter applies to multi-step simulations run via GEMSIM or a TABLO-generated program, or to Johansen simulations run via SAGEM. Section 5.4 also applies to other programs such as GEMPIE where you are choosing a set of variables from a larger set.

5.1 Specifying the Closure

In the economic models solved via GEMPACK, the number of variables[29] exceeds the number of equations (as you have seen in section 2.13.1 of GPD-1). In order to solve the equations, some of the variables must be predetermined – the **exogenous** variables – by giving them actual numerical values. This is done by specifying the shocks. From these shocks, the equations of the model can be solved to determine the remaining variables – the **endogenous** variables.

The **closure** of the model, or alternatively the economic **environment** in which you are carrying out a simulation, is the partition (split) of the set of variables into exogenous and endogenous.

For the closure to be valid, the number of endogenous variables must be equal to the number of equations. In addition, the exogenous variables must be an independent set with no equations between them. For example, consider a set of (scalar) variables x, y, and z with an equation between them

$$x + y + z = 0.$$

Given x and y as exogenous variables, z cannot also be chosen as exogenous since it is already determined by the equation

$$z = - x - y.$$

See section 5.2.7 below for more about this.

The actual choice of exogenous variables depends on what the simulation represents. In general terms variables determined by factors outside the system are often made exogenous, and the remaining variables which are determined by the system being modelled are the endogenous variables.

We recommend that you use GEMPACK Command files for carrying out simulations with GEMSIM, TABLO-generated programs or SAGEM.[30] In section 5.2 below we discuss the different ways of specifying the closure on a Command file.

If you have levels VARIABLEs in your TABLO Input file, you can use the levels names for variables in Command files for GEMSIM or TABLO-generated programs, but not for SAGEM[31]. Alternatively you can use the names of the associated linear variables (these often have "p_" or "c_" added at the start – see section 2.2 of GPD-2) when giving names to specify the closure (or when you choose other sets of variables).

---

[29] As explained in section 2.13.1 of GPD-1, here the number of variables is obtained by adding up the number of components in each of the vector variables. (The VARIABLEs in the TABLO Input file are these vector variables; each has a number of components determined by the sizes of the sets over which their arguments range.)

[30] We recommend that you never try to run TABLO-generated programs, GEMSIM or SAGEM interactively, but always use a Command file.

[31] This was introduced in Release 8.0 - see section 5.10.

5.1.1 Miniature ORANI Model

Throughout this chapter, examples used are taken from a version of the Miniature ORANI model documented sections 3-9 of Dixon *et al* (1982). For this chapter, the only things you need to know about this model are the names of the sets in this model and the names of the variables in this model, and the number of components each has, as shown below.

Note that in the TABLO Input file (MO.TAB), all variables are levels variables, for example XHOUS. Associated linear variables have the prefix "p_" or "c_", for example p_XHOUS.

SETS

Name	Size	Description	Element names
COM	2	commodities	c1,c2
IND	2	industries	I1,I2
SOURCE	2	source of commodities	domestic,imported
FAC	2	primary factors	labor,capital

VARIABLES

Name	Number of components	Arguments (if any)	Description
XHOUS	4	(COM,SOURCE)	household consumption
CHOUS	1	-	nominal total household consumption
PCOM	4	(COM,SOURCE)	commodity prices
PDOT	2	(COM)	unit cost of Armington aggregate
XDOT	2	(COM)	Armington aggregate over source
U	1	-	consumption mix ratio (household)
PEXP	2	(COM)	export prices (foreign dollars)
PIMP	2	(COM)	import prices (foreign dollars)
XEXP	2	(COM)	export demands
FEXP	2	(COM)	export demand shifters
Z	2	(IND)	industry activity
YCOMIND	4	(COM,IND)	industry output
XINTCOM	8	(COM,SOURCE,IND)	intermediate commodity inputs
XINTFAC	4	(FAC,IND)	intermediate factor inputs
PLAB	1	-	wage rate
PCAP	2	(IND)	price of capital
PFAC	2	(FAC,IND)	price of primary factors
XINTCOM_CD	4	(COM,IND)	Cobb-D combination of inputs com
PCOM_CD	4	(COM,IND)	price of Cobb-D combination of com
XINTFAC_CD	2	(IND)	Cobb-D combination of factors
PFAC_CD	2	(IND)	price Cobb-D combination factors
V	2	(COM)	power of export subsidy
T	2	(COM)	power of import duty
XLAB	1	-	total demand for labor
XCAP	2	(IND)	industry demand for capital
M	1	-	total imports (foreign or overseas value)
E	1	-	total exports (foreign or overseas value)
CPI	1	-	consumer price index
FWAGE	1	-	wage rate shifter
CR	1	-	real household consumption
XIMP	2	(COM)	import quantities
INTCOM	8	(COM,SOURCE,IND)	intermediate input of commodities
INTFAC	4	(FAC,IND)	intermediate input of factors
HOUSE	4	(COM,SOURCE)	household consumption of commodities
EXPCOM_DOMV	2	(COM)	exports of (domestic) commodities
IMPCOM_DOMV	2	(COM)	imports of commodity (Aust $)
COMPROD	4	(COM,IND)	production of commodity I by ind j
PHI	1	-	exchange rate
B_A	1	-	Aust dollar change in trade balance
B_F	1	-	foreign dollar change in trade balance
EXPSUB	2	(COM)	export subsidies on exports of commodities
DUTY	2	(COM)	duty levied on imports of commodities

If you want to know more about the model, note that the TABLO Input file MO.TAB for this model is included in the model files which accompany GEMPACK – see chapter 1 of GPD-8. To run Miniature ORANI, run TABLO with the TABLO Input file MO.TAB, in the usual way as described in chapter 2 of GPD-1. You can then run the program MO.EXE or GEMSIM using the Command file MOTAR.CMF (which is also included in the model files examples) to carry out a tariff simulation with this version of the model.

5.2 Specifying the Closure via a Command File

The three main ways of specifying the closure using a Command file are

1. starting from scratch by giving the closure as a list of variable names and components,
2. starting from an existing closure saved on a file and using this closure unchanged, or
3. starting from an existing closure saved on a file and modifying it.

It is also possible to use data files to specify which components of a variable are exogenous or endogenous, as explained in section 5.2.5.

5.2.1 Specifying the Closure from Scratch

The simplest way of specifying a closure is to give a list of exogenous variables and then specify that the rest of the variables are endogenous. The Command file statements for this are

```
exogenous <list> ;
rest endogenous ;
```

where the <list> is a list of variable names, optionally followed by set/element arguments or component numbers separated by spaces. We give examples below. More details about what is allowed in <list> can be found in section 5.4.1 below. The <list> contains names of variables either linearized or levels variables[32] – see section 5.10 about the use of levels variables in Command files. However you cannot use levels variable names in Command files for SAGEM, only linearized variable names.

- Individual component numbers can follow the variable names; for example "p_XINTCOM 2-4 6 8" would mean components 2,3,4,6,8 of p_XINTCOM, the percentage change in the variable XINTCOM. See section 5.3 below for details about component numbers for variables (such as p_XINTCOM) with more than one argument.
- Alternatively, components can be indicated using set and/or element names[33] where available; for example, p_XINTFAC("labor",IND) would specify all components of p_XINTFAC with first argument "labor" and second argument in the set IND (that is, all labor inputs to all industries – these are components numbered 1 and 3 of p_XINTFAC - see section 5.4.1 below).
- If a variable name has no set/element arguments and no component numbers following it, this means that all components are being set exogenous.

[32] Prior to Release 8.0 of GEMPACK these were the names of the associated linearized variables at this stage, since TABLO has already linearized any levels equations into expressions containing percentage changes or ordinary changes of levels variables, as explained in section 2.2 of GPD-2.

[33] Subset names can also be used. In the example that follows, if there were a subset IND2 of IND declared in the TABLO Input file, then you could write p_XINTFAC("labor",IND2) to indicate all components of INTFAC with first argument "labor" and second argument in IND2.

For example, the following statements specify the usual closure for Miniature ORANI:

```
exogenous p_PIMP p_FEXP p_XEXP("c2") p_PHI
      p_V 1 p_FWAGE p_CR p_T p_XCAP ;
rest endogenous ;
```

Conversely you can list the endogenous variables and make the rest exogenous as in the following statements:

```
endogenous p_XHOUS p_CHOUS p_PCOM p_PDOT p_XDOT p_U ;
endogenous p_PEXP p_XEXP("c1") ;
endogenous p_Z p_YCOMIND p_XINTCOM p_XINTFAC ;
endogenous p_PLAB p_PCAP p_PFAC p_XINTCOM_CD p_PCOM_CD ;
endogenous p_XINTFAC_CD  p_PFAC_CD ;
endogenous p_V 2 p_XLAB p_M p_E p_CPI p_XIMP ;
endogenous p_INTCOM p_INTFAC p_HOUSE p_EXPCOM_DOMV ;
endogenous p_IMPCOM_DOMV p_COMPROD ;
endogenous c_B_A c_B_F c_EXPSUB c_DUTY ;
rest exogenous ;
```

This would give exactly the same closure as in the first method.

Note that you can continue one statement over several lines (as in the "exogenous …" statement in the first example above). You can also have several "exogenous …" or "endogenous …" statements (as in the second example above).

If you want to **save** this closure on an **Environment file** so that you can use it later in another simulation, the command

`save Environment file mo ;`

can be included. This saves an Environment file MO.EN4 containing the closure specified. The Environment file suffix (.EN4) is added by the software. Your Environment file name in the Command file should not have a suffix.

You can use XSET and XSUBSET statements (see section 6.6) when specifying the closure. To see an example, look at the ORANIG Command file OG01WAGE.CMF (see section 1.4.1 of GPD-8) which specifies the short-run closure for ORANIG.

Excerpt from Command file OG01WAGE.CMF

```
! Distribution of investment between industries
xSet EXOGINV  # 'exogenous' investment industries #
 (Utilities, Construction, Finance, Dwellings, PublicServcs, PrivatServcs);
xSubset EXOGINV is subset of IND;
xSet ENDOGINV # 'endogenous' investment industries # = IND - EXOGINV;
exogenous finv1(ENDOGINV); ! investment linked to profits
exogenous finv2(EXOGINV);  ! investment follows aggregate investment
```

The set EXOGINV is not mentioned in the TAB file ORANIG01.TAB, only in the Command file. If you changed the industries in the set EXOGINV, this would change the closure.

5.2.2 Using a Saved Closure

If the closure used in a simulation has been saved on an Environment file, this file can then be used to specify the closure for a new simulation by using the command in the Command file

`use Environment file <filename> ;`

For example, using the Environment file MO.EN4 from the example above, you can reuse exactly the same closure by the command

`use Environment file mo ;`

The closure is also written on the Solution file. This file can be used to give the closure instead of the Environment file. For example,

```
take closure from Solution file <filename> ;
```

is one way of specifying the closure.[34]

However the closure for a model cannot be taken from a Solution file when any backsolving has been carried out. This is because, when the closure is being processed, only variables in the condensed system for the model are relevant. (As far as the closure-processing part of the software is concerned, these backsolved-for variables are not in the model.) Thus, for example, the statement

```
take closure from Solution file <filename> ;
```

cannot be used in a Command file when this Solution file contains the values of variables which have been backsolved for.

5.2.3 Modifying a Saved Closure

If in a simulation you want to have a closure which is very similar to another closure which you have saved, you can start with the old closure and modify it as in the following example from Miniature ORANI.

```
Modify closure from Environment file mo ;
swap p_PHI = p_CPI ;
swap p_XEXP("c2") = p_V("c2") ;
```

This starts with the closure on file MO.EN4. In this closure as given in the first example in section 5.2.1 above, p_PHI is exogenous and p_CPI is endogenous. The first swap statement makes p_CPI exogenous and p_PHI endogenous. Similarly the second swap statements makes the previously exogenous component p_XEXP("c2") to be endogenous and the previously endogenous component p_V("c2") to be exogenous.

Another example is the swapping of several components of one variable with several components of another variable as in the following:

```
modify closure from Environment file mo ;
swap p_PIMP 1 2 = p_PCOM 1 4 ;
```

Note that, in a "swap" statement, if all components on one side are exogenous, those on the other side must all be endogenous (and vice versa). The total number of components on each side must be the same. These components are interchanged by "swapping" the sets to which they belong.

You can also use "exogenous … ;" and "endogenous … ;" statements to modify a closure. For example, the following statements would specify the same modified closure as those in the first example above.

```
Modify closure from Environment file mo ;
endogenous p_PHI p_XEXP("c2") ;
exogenous p_CPI p_V("c2") ;
```

A summary of the syntax for these commands and other similar commands is given in chapter 18. Examples of complete Command files are given in sections 2.8.1 and 2.12.4 of GPD-1 and in sections 18.1.2 and 18.2.1 of this document and in the model files listed in chapter 1 of GPD-8.

5.2.4 Order of Command File Statements Can Be Important

Usually the order of statements on a Command file can be changed without any effect on the closure. For example, changing the order of the statements

```
exogenous p_V;
exogenous p_XINTFAC ;
```

would not affect the closure. However, occasionally the order of statements can affect the result and/or the validity of the commands, as the following example (not relating to Miniature ORANI) makes clear:

[34] You can also take, or modify, the closure from an LU file (see section 9.3).

```
exogenous y ;
endogenous y 1 ;
swap y 1 = z 5 ;
```

5.2.5 Closure Specified Using Data Files

It is also possible to use data on a text file to specify which components of one or more variables are to be exogenous or endogenous.[35] For example, consider the statement

```
exogenous p_xfac = nonzero value on file sjclos.dat ;
```

Here the file SJCLOS.DAT should be a GEMPACK text data file (see chapter 6 of GPD-4) with one array of real data (not integer data). The correct size of the array is that needed to hold all of the components of variable p_xfac, both exogenous and endogenous. This statement means that those components of p_xfac which correspond to nonzero values on the file SJCLOS.DAT will be set exogenous. For example, if the model is Stylized Johansen, when p_xfac has two components in the set SECT, and if SJCLOS.DAT is

```
2 real ;
 0  5
```

then just component number 2 of p_xfac will be set exogenous.

In general, statements of the form

exogenous | endogenous <variable> = zero | nonzero | positive | negative value on file <file> ;

are allowed.

The "how much data" information must be the same as on a shock file (see section 5.8) for the variable. If, for example, the variable has two arguments ranging over sets of sizes 4 and 6 respectively, the how much data information can be either

```
24 ;
or
4  6 ;
```

Each file can only contain one array of data. But the same file can be accessed more than once in a Command file as, for example, in

```
exogenous x1 = nonzero value on file DAT1.DAT ;
endogenous x1 = zero value on file DAT1.DAT ;
```

[35] This was introduced in Release 5.2. This method of specifying part of the closure was introduced to facilitate the closure swaps needed when handling quotas and other inequalities via the (now superseded by chapter 16) methods in Bach and Pearson (1996).

5.2.6 List of Exogenous/Endogenous Variables when Closure is Invalid

If the closure specified on a Command file is not valid because of an incorrect number of exogenous variables, GEMSIM or the TABLO-generated program echoes the lists of exogenous and endogenous variables. These variables are grouped according to argument type.[36] [For example, all scalar variables are first. Then all variables with a single argument ranging over the set COM. And so on.]

The number of condensed equations with the same argument type is given for each type. This may assist in identifying the closure problem since, in many closures, the number of endogenous variables with a certain argument type equals the number of equations with this argument type.

An example will make this clearer.

Example. Suppose that we (incorrectly) give the closure for Stylized Johansen via

```
exogenous p_XFAC p_XC ;
rest endogenous ;
```

The LOG file from GEMSIM or the TABLO-generated program will be as shown in Table **5.2.6** on the next page. [We have marked the start of the different argument types in bold.]

The SECT group shows the pattern in most closures. There are 4 equations with this argument type and 4 variables with this argument type all of whose components are endogenous.

The software identifies at the end the argument types which do not satisfy this simple pattern and suggests that the problem may lie in one or more of these argument types. Notice that the scalars and NUM_SECT groups do not satisfy this simple pattern (even though there is no closure problem with these argument types). However, the software correctly identifies the SECT*SECT group as a possible source of error.

5.2.7 Initial Closure Check and Singular Matrices

The program checks the closure quite early – usually soon after all the sets and subsets have been processed (see section 6.2.1). This initial closure check is only a check that the number of endogenous variables is equal to the number of equations.

This initial check does not guarantee that it is actually possible to solve for all the endogenous variables. That check (which is also a check that the exogenous variables are truly independent) is only done when the LU decomposition (see section 12.1) of the LHS Matrix is calculated. At that stage, it may be discovered that it is not possible to solve the model for all the endogenous variables because the LHS Matrix is singular (see section 15.1).

You can ask that just this initial check be done – see section 5.9 below.

[36] This grouping was introduced for Release 8.0.

```
  [ Closure is not yet valid.
    Current closure : 6 exog, 23 endog.
    You require     : 2 exog, 27 endog.  ]
    (You require 4 more endogenous.)

   Current Closure on Command file:
    [Variables are grouped according to their set arguments.]
    [Only variables and equations in the condensed system
      are mentioned in this report.]

! SCALARS (ie MACROS)
!  0 equations
!  1 variables with ALL components endogenous
 ?? Possible closure problem here ??
Endogenous
p_Y
 ;

! SECT  [Size: 2.]
!  4 equations
!  4 variables with ALL components endogenous
Endogenous
p_PC
p_XCOM
p_XH
p_DVHOUS
 ;

! FAC  [Size: 2.]
!  1 equations
!  1 variables with ALL components endogenous
Endogenous
p_PF
 ;
Exogenous
p_XFAC
 ;

! NUM_SECT  [Size: 1.]
!  1 equations
!  0 variables with ALL components endogenous
 ?? Possible closure problem here ??

! SECT*SECT  [Size: 2x2=4.]
!  2 equations
!  1 variables with ALL components endogenous
 ?? Possible closure problem here ??
Endogenous
p_DVCOMIN
 ;
Exogenous
p_XC
 ;

! FAC*SECT  [Size: 2x2=4.]
!  2 equations
!  2 variables with ALL components endogenous
Endogenous
p_XF
p_DVFACIN
 ;

  There may be a problem with the following 3 argument types:
  SCALARS (ie MACROS)
  NUM_SECT
  SECT*SECT
 Of course there may be problems with other argument types.
   [See section 5.2.6 of GPD-3 for more details.]

  End of Closure on Command file.
```

Table 5.2.6 : Example Output when Closure is Invalid

5.3 Component Numbers for a Variable

You will need to know the order of the components of a variable if you want to specify component numbers in the statements giving the closure (as above) or in selecting sets of variables (section 5.4 below) or in specifying shocks (section 5.5 below).

A VARIABLE with no arguments has just one component. A variable with one argument (for example, p_PEXP(i) in Miniature ORANI) has as many components as the size of the set over which the argument runs; and these components are in the same order as the elements of this set.

For variables with 2 or more arguments, the order of the components is determined by the rule that the first argument varies fastest, then the second varies next fastest, and so on.

The examples below (all taken from Miniature ORANI as described in section 5.1.1 above) should make this clear.

5.3.1 Examples of Component Numbers

p_XEXP has one argument ranging over the set COM.

```
Component 1     p_XEXP("c1")
Component 2     p_XEXP("c2")
```

p_YCOMIND has two arguments ranging over COM and IND.

```
Component 1     p_YCOMIND("c1","I1")
Component 2     p_YCOMIND("c2","I1")
Component 3     p_YCOMIND("c1","I2")
Component 4     p_YCOMIND("c2","I2")
```

p_XINTCOM has three arguments ranging over COM, SOURCE and IND respectively. The first 4 components correspond to the first industry "I1" and the last 4 to the second industry "I2" since the last argument varies slowest.

```
Component 1     p_XINTCOM("c1","domestic","I1")
Component 2     p_XINTCOM("c2","domestic","I1")
Component 3     p_XINTCOM("c1","imported","I1")
Component 4     p_XINTCOM("c2","imported","I1")
Component 5     p_XINTCOM("c1","domestic","I2")
Component 6     p_XINTCOM("c2","domestic","I2")
Component 7     p_XINTCOM("c1","imported","I2")
Component 8     p_XINTCOM("c2","imported","I2")
```

5.4 Choosing Other Sets of Variables

There are many situations in which you must choose a set of variables. For example,

1) when carrying out a multi-step simulation, you must choose the endogenous variables to be retained on the Solution file (the so-called "cumulatively-retained endogenous variables"), or
2) when printing totals results via GEMPIE, you must choose the endogenous variables to be in the printout.

In such situations, there is always a "big" set and you are choosing a subset of this big set. In example (1) above, the big set is the set of all endogenous variables (including any backsolved for if you condensed the model) while in (2) above it is the set of cumulatively-retained endogenous variables.

In GEMSIM, TABLO-generated programs and SAGEM, you can choose sets of variables using a Command file. This method is illustrated in section 5.4.1.

In choosing sets of variables in other programs, in particular GEMPIE, Command files are not available so you may need to choose sets of variables interactively (or by using a Stored-input file). Some details about this are given in section 5.4.3 below. Complete details of interactive choice of sets of variables are given in chapter 17 of GPD-4.

Prior to Release 8.0 of GEMPACK, if you had levels VARIABLEs in your TABLO Input file, you had to use the names of the associated linear variables (these often have "p_" or "c_" added at the start – see section 2.2 of GPD-2) when choosing sets of variables. Now you can use either the name of the associated linear variable or the levels name in your Command file for GEMSIM or TABLO-generated programs. For SAGEM you cannot use the levels name, only the linear variable name.

5.4.1 Choosing Sets of Variables via a Command File

There are two choices of sets of variables for **GEMSIM or TABLO-generated programs**, namely:

(i) Cumulatively-retained endogenous variables which are chosen from the set of all endogenous variables (including any backsolved for if you condensed the model),

(ii) The set of variables which are on the Extrapolation Accuracy file (that is, the XAC-retained variables), chosen from the set of all endogenous variables (including any backsolved for if you condensed the model).

For **SAGEM**, there are various choices to make, including:

(a) Individually-retained exogenous variables from the set of all shocked exogenous variables.

(b) Individually-retained endogenous variables from the set of all endogenous variables.

(c) Cumulatively-retained endogenous variables from the set of all endogenous variables.

The meanings of individually-retained and cumulatively-retained are explained in detail in chapter 10. Here we will just assume that in each choice, you have a "big" set and are choosing a subset of this set. The Command file statements are similar for all these choices so we will give them all together and illustrate by examples the form of the <list> used in specifying them:

```
cumulatively-retained endogenous <list> ;
XAC-retained <list> ;
individually-retained exogenous <list> ;
individually-retained endogenous <list> ;
```

In each case, the <list> must consist of variables and components in the "big" set. The things in the <list> can be

1) a variable name followed by set and/or element arguments, for example [37]

p_XINTFAC("labor",IND)

[37] There must not be a space between the end of the variable name and the '('. For example, "p_PEXP(COM)" and "p_PEXP (COM)" mean quite different things - see (5) below for the meaning of "(COM)" in the second case.

which means all components of p_XINTFAC with first argument "labor" and second argument in the set IND (that is, all labor inputs to all industries – these are components numbered 1 and 3 of p_XINTFAC). This means that the components indicated are to be included in the subset. (They must all be in the "big" set.)

2) a variable name followed by certain component numbers (these components must all be in the big set), for example,

   p_XINTCOM 2-4 6

   which means that just these components are to be included in the subset. [See section 5.3 to see how component numbers are worked out.]

3) a variable name not followed by set/element arguments or component numbers, which means that all components of this variable which are in the big set are to be included in the subset.

4) abbreviations such as

%all	meaning all variables in the big set, or
%macro or **%scalar**	meaning all macro or scalar variables in the big set are to be included in the subset. [A "macro" or "scalar" variable is, by definition, one with just one component, such as p_CHOUS in Miniature ORANI.]

5) abbreviations such as

(FAC)	meaning that all variables with one argument ranging over the set FAC which are in the big set are to be included in the subset, or
(COM,SOURCE)	meaning the same for variables with two arguments, the first ranging over COM and the second ranging over SOURCE.

5.4.2 Example of Choosing Sets in SAGEM

Suppose shocks are to be applied to the following exogenous variables from Miniature ORANI. [We are assuming the closure is as saved on the Environment file MO.EN4 – see the section 5.2.1. We are also assuming that an Equations file MO.EQ4 has been created (for example, by running GEMSIM via Command file MOTAR.CMF supplied with the GEMPACK examples).]

p_PHI, component 2 of p_T, p_FWAGE and p_CR.

We show the Command file (called **MOSAGEM.CMF** as supplied with the GEMPACK examples) below.

The three commands in the Command file below choosing sets of variables determine what is on the Solution file and hence what you can print out later using GEMPIE. The first two commands – see (1) and (2) below – refer to individual solutions on the Solution file while command (3) refers to the totals solution.

```
! Command file for running SAGEM for the Miniature ORANI model.
! It relies on an Equations file MO.EQ4 and an
!  Environment file MO.EN4. [Both of these are produced
!  when a simulation is run with MOTAR.CMF]

use equations file  mo ;
use environment file mo ;

! Name of Solution file is inferred from name of Command file.
! (See section 2.5 of GPD-3.)

! Shocks
shock p_T 2 =  1 ;   ! or 'shock p_T("c2") = 1 ;'
shock p_PHI = 1 ;
shock p_FWAGE = -2.38 ;
shock p_CR = 2.76 ;
!
! Choosing sets of variables
individually-retained exogenous %all ;
individually-retained endogenous p_Z p_YCOMIND(COM,"I2")
                                 p_XINTFAC 1-3 %macro ;
cumulatively-retained endogenous p_Z p_YCOMIND (COM) (FAC,IND) ;

! Subtotals results
subtotal p_T = tariff shock ;
subtotal p_PHI = exchange rate shock ;
subtotal p_FWAGE p_CR = wage and real consumption shocks ;

!
verbal description = MO Standard closure.
Shocks  p_phi p_t 2 p_fWAGE p_cR ;
```

(1) The command

```
individually-retained exogenous %all ;
```

means that the Solution file will show results for all shocks. There will be a separate column for each of the shocks (see section 10.1.1). (The big set here is the set of all shocked variables.)

(2) The command

```
individually-retained endogenous p_Z  p_YCOMIND(COM,"I2")
      p_XINTFAC 1-3 %macro ;
```

means that the Solution file will contain the results of each of these shocks on

variable p_Z (all components),

components (c,"I2") for all 'c' in COM of p_YCOMIND,

components 1,2,3 for variable p_XINTFAC, and

all endogenous macro variables (that is, variables with just one component), namely
p_CHOUS, p_U, p_PLAB, p_XLAB, p_M, p_E, p_CPI, c_B_A, c_B_F .

(The big set here is the set of all endogenous variables.)

(3) The command

```
cumulatively-retained endogenous p_Z  p_YCOMIND  (COM)  (FAC,IND) ;
```

means that the Solution will contain the cumulative effect of all the shocks on the following.

All components of variables p_Z and p_YCOMIND.

(COM) chooses all (endogenous) components of all variables
in the model having one argument ranging over the set COM, namely

p_PDOT, p_XDOT, p_PEXP, p_XEXP, p_V, p_XIMP,
p_EXPCOM_DOMV, p_IMPCOM_DOMV, c_EXPSUB and c_DUTY.

(FAC,IND) chooses all variables in the model having two
arguments ranging over the sets FAC and IND (in that order), namely

p_XINTFAC, p_PFAC and p_INTFAC.

(The big set here is the set of all endogenous variables.)

(4) This Command file also sets up (and stores on the Solution file) three subtotals results. See sections 10.2 and 10.2.1 for more details.

To see the effects of these statements, see the hands-on example using this Command file in section 2.5.6 of GPD-8.

5.4.3 Choosing Sets of Variables Interactively

Where Command files are available, we strongly recommend that you always use them since making choices interactively (or even on a Stored-input file) is less reliable and less transparent.

In choosing sets of variables in other programs, in particular GEMPIE and SLTOHT, Command files are not available so you may need to choose sets of variables interactively (or by using a Stored-input file).

You will recognise the choice situation because you will be presented with a menu looking something like that shown below.[38] (In the menu below, the words "shocked exogenous" and "individually-retained exogenous" will be replaced, more generally, by words describing the big set and the subset being chosen.)

```
     CHOICE OF WHICH shocked exogenous VARIABLES YOU WANT
                  TO BE individually-retained exogenous.
  Make ONE of the following choices:
   L     LISTS of variables, all or some of whose
          components are to be individually-retained exogenous.
   A     ALL components of ALL shocked exogenous
          variables to be individually-retained exogenous.
   N     NO shocked exogenous
            variables to be individually-retained exogenous.
   M     All shocked exogenous MACRO
            variables to be individually-retained exogenous.
   1     All components of ONE shocked exogenous
          variables to be individually-retained exogenous.
   F     All components of a FEW shocked exogenous
          variables to be individually-retained exogenous.
   S     SOME components of SOME shocked exogenous
          variables to be individually-retained exogenous.
   W     WHICH are the shocked exogenous variables.
  Enter your choice now. ('L' is the default.)
```

Example of the Menu Presented for Choosing a Subset

Full details about choosing subsets interactively in these situations can be found in chapter 17 of GPD-4.

[38] This is the menu if you are using full prompts. If you have chosen brief prompts, the menu will be compressed considerably.

5.5 Specifying the Shocks

On your Command file, you specify the variables to be shocked and the numerical values of the shocks at the same time, as explained in section 5.5.1 below. [39]

The general forms of a shock statement on Command files are

```
shock <variable-name> = ... ;
shock <variable-name> <some-components> = ....;
```

After the '=' sign there is a list of values, a file name or various other ways of supplying a list of shock values to be applied to the shocked variable.

Only variables with at least some components exogenous can have shock statements. The first statement above (which has no component numbers or arguments after the variable name) means that ALL EXOGENOUS components of the variable are to be shocked. The information after the '=' sign must be consistent with this (correct number of shocks etc).

Consider, for example, the statements

```
Exogenous p_XINTFAC("capital",IND) ;
Shock p_XINTFAC = 4 6;
```

Here only two components of variable p_XINTFAC (see section 5.1.1) are shocked.

Note that the shocks to a particular variable are said to be **uniform** if all shocked components of this variable are given the same numerical shock.

5.5.1 Specifying the Shocks via a Command File

To specify the shocks to exogenous variables using a Command file, you must have (at least) one shock command for each variable to be shocked. All or some components of this variable can be shocked. In Command files for GEMSIM and TABLO-generated programs you can use linear or levels variable names[40] but only linear variable names are allowed in Command files for SAGEM.

If only a few shocks are involved, a simple list of components and corresponding shocks can be given. One example of this type of command is given in the Command file in section 5.4.1.

```
! Shocks
!
shock p_T 2 =  1 ;  ! or 'shock p_T("c2") = 1 ;'
shock p_phi = 1 ;
shock p_fWAGE = -2.38 ;
shock p_cR = 2.76 ;
```

You can specify shocks to several numbered components of a variable. For example if p_XINTFAC were exogenous and you wanted to shock components 1, 2 and 4, the command

```
shock p_XINTFAC 1 2 4 = 0.2 0.4 –3.9 ;
```

means that component 1 of XINTFAC is shocked by 0.2, component 2 by 0.4 and component 4 by –3.9. [See section 5.3 to see how component numbers are worked out.]

Alternatively the components to shock can be indicated using sets and/or elements as arguments. For example,

```
shock p_XINTFAC("capital",IND) = 4 6 ;
```

means shock p_XINTFAC("capital","I1") by 4 per cent and shock p_XINTFAC("capital","I2") by 6 per cent.

---

[39] If you are running interactively (which we do not recommend), you could specify the variables to be shocked following the procedure described in section 5.4.3 above and then specify the numerical values of the shocks separately, as explained in section 5.12 below.

[40] Levels names were not allowed prior to Release 8.0.

For a uniform shock applied to just components 1, 2, 4 of p_XINTFAC, the command is

```
shock p_XINTFAC 1 2 4 = uniform 3.0 ;
```

or to shock just components relating to capital inputs uniformly,

```
shock p_XINTFAC("capital",IND) = uniform 3.0 ;
```

or, if all components of XINTFAC are shocked uniformly,

```
shock p_XINTFAC = uniform 3.0 ;
```

You can use XSET and XSUBSET statements (see section 6.6) when specifying shocks. To see this, look at the Command file **GSEFTA1.CMF** (supplied with the GEMPACK examples – see section 1.6.1 of GPD-8). This Command file contains the statements

```
! Shocks
XSet AG_COMM # Agricultural commodities # (agric, food) ;
XSubset AG_COMM is subset of TRAD_COMM   ;
XSet NONAG_COMM = TRAD_COMM - AG_COMM ;
Shock tms(NONAG_COMM, "safrica", "eunion") =
      select from file gtmsa7x5.shk ;
```

The sets AG_COMM and NONAG_COMM are used when specifying the shocks. See section 5.5.2 below for the meaning of "select from". [In this simulation, the European Union "eunion" is only eliminating import tariffs on non-agricultural commodities imported from South Africa "safrica". Here the file gtmsa7x5.shk contains the shocks needed to completely eliminate all import tariffs on all commodities for all regions.][41]

5.5.2 Using a Shock File

When you are giving a large number of shocks to a variable, you can use a **shock file**.
For example,

```
shock p_XINTFAC = file FACTOR.SHK ;
```

tells the program to read the shocks from the text file FACTOR.SHK into the components of variable p_XINTFAC.

Shock files can be text files or Header Array files.[42] When shocks are read from a Header Array file, the header must be included in the statement as well as the file name. For example

```
shock p_XINTFAC = file FACTOR2.SHK Header "XIFC" ;
```

tells the program to read the shocks from the value at header "XIFC" on the Header Array file FACTOR2.SHK into the components of variable p_XINTFAC. If you include "Header" in the shock statement, the program assumes that the file is a Header Array file. If the statement does not include "Header" the program assumes that the file is a text file.

The syntax for text shock files is described near the end of this section. Section 5.8 below contains fine print details about checking the sizes of arrays on a shock file and how data on the file is associated with the shocked components of the variable being shocked.

You can read shocks for several variables from the same Header Array shock file since you can use different Headers for different variables.

Each text shock file should only contain one array of data. However the same text shock file can be accessed more than once in a simulation, as in, for example,

```
Shock x1 = file x1.shk ;
Shock x2 = file x1.shk ;
```

---

[41] Under RunGTAP, the XSET and XSUBSET statements here are included in the CMFSTART file in directory ASA7X5. [See the RunGTAP Help file for information about CMFSTART files.]

[42] Header Array shock files were introduced in Release 7.0.

There are situations in which you have computed on a file (perhaps a file written by GEMSIM or a TABLO-generated program) possible shocks to all components of a variable, but only want to apply some of these shocks in the current simulation. In such a case, you do not have to edit the file to remove the values corresponding to the non-shocked components; instead you can direct the program to select the values on the file corresponding to the components of interest. For example, the statement

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

will shock only the two components ("capital","I1") and ("capital","I2") of p_XINTFAC and read the relevant values from the file FACTOR.SHK. In this case, the text file FACTOR.SHK must have 4 values on it, and the second and fourth are read (since the components mentioned are components number 2 and 4 of variable p_XINTFAC – see section 5.4.1 above). The words "**select from file**" are what distinguish this from other cases.

Similarly you can use "select from" followed by a list of values, as in, for example,

```
shock p_XINTFAC("capital",IND) = select from 3 4 5 6 ;
```

Here components ("capital","I1") and ("capital","I2") of XINTFAC will be shocked by 4 and 6 per cent respectively.

If you have shocks to a large number of variables written by another program (perhaps a TABLO-generated one), it may be more convenient to put all shocks onto a single Header Array file rather than separate text files for each variable.

There are advantages in using a Header Array file where the set elements are labelled since this makes it easy to check which shock is applied to which set element. Using ViewHAR it is easy to make a Header Array of the correct size containing the default value of, for example, 0.0. You could then paste the shocks you want to use in the correct positions. For two or more dimensional arrays, this avoids counting components and makes checking easy.

If you have a file containing the closure (for example an Environment or Solution file) and a SOL file (a Header Array file produced from a Solution file by SLTOHT), program SEENV can be used to write one of these shock statements for each exogenous variable on the Environment file. See section 12.3 of GPD-4 for details.

If you are shocking only some components of a variable, we suggest the following.

- We believe that, if you need to specify components, using arguments (rather than component numbers) is the most easily understood.
- We think that shock statements containing component numbers can sometimes be difficult to understand.
- Remember that you can use "select from file" in complicated cases to make the correspondence between the shocked components and the shocks on the file clear.

Examples using Shock Files

Below are a few examples of shock statements using a shock file. In some cases we have used a Header Array file ALLSHOCKS.HAR to hold the shocks.

The variable we are shocking is the COM x SRC x IND variable **a1csi,** of size 15 x 2 x 13, as defined in the excerpt from the TABLO Input file below.

Excerpt from TABLO Input file

```
Set COM (com1-com15) ;
Set MARG (com10, com14) ;
Subset MARG is Subset of COM ;
Set SRC (dom, imp) ;
Set IND (ind1-ind13) ;

Variable (All,c,COM)All,s,SRC)(All,i,IND) a1csi(c,s,i) ;
```

The examples below illustrate various possible cases.

1. All components exogenous and all components shocked:
   exogenous a1csi ;
   shock a1csi = file ALLSHOCKS.HAR header "1ALL" ;
   The Header array "1ALL" contains a three-dimensional COM x SRC x IND array with dimensions (15, 2, 13). All components of a1csi are shocked.

2. All components exogenous and some components shocked:
   exogenous a1csi ;
   shock a1csi(COM,"dom",IND) = select from file ALLSHOCKS.HAR header "1ALL" ;
   The Header array "1ALL" contains a three-dimensional COM x SRC x IND array with dimensions (15, 2, 13) but only the "dom" shocks are applied.
   or
   shock a1csi(COM,"dom",IND) = file ALLSHOCKS.HAR header "1DOM";
   The Header "1DOM" contains a three-dimensional COM x "dom" x IND array with dimensions (15, 1, 13) or a two dimensional COM x IND array with dimensions (15,13). All shocks in header "1DOM" are applied to the "dom" components of a1csi.
   or
   shock a1csi("com1","dom",IND) = file ALLSHOCKS.HAR header "COM1";
   The Header "COM1" contains a one-dimensional IND array with dimension (13).

3. Some components exogenous and the same components are shocked:
   exogenous a1csi(COM,"dom","ind1") ;
   shock a1csi = file a1some.shk ;
   a1some.shk is a text file containing a one-dimensional COM array with 15 components.
   Note that only the exogenous components are shocked (see section 5.11.1). The shock statement here is equivalent to **shock a1csi(COM,"dom","ind1") = file a1some.shk ;**
   or
   exogenous a1csi(COM,SRC,"ind1") ;
   shock a1csi = select from file ALLSHOCKS.HAR header "1ALL" ;
   The Header array "1ALL" contains a three-dimensional COM x SRC x IND array with dimensions (15, 2, 13), but only the 15 x 2 shocks for industry "ind1" are applied.

4. Some components exogenous and a subset of the components shocked:
   exogenous a1csi(COM,"dom",IND) ;
   shock a1csi(MARG,"dom",IND) = select from file ALLSHOCKS.HAR header "1ALL";
   The Header array "1ALL" contains a three-dimensional COM x SRC x IND array with dimensions (15, 2, 13). Shocks are applied to the 2 x 13 components (m,"dom",i) where m is in the subset MARG of COM and i is in the set IND.
   or
   exogenous a1csi(COM,"dom",IND) ;
   shock a1csi("com1","dom",IND) = file a1com1.shk ;
   a1com.shk is a text file containing a one-dimensional IND array with 13 components.

Syntax of Text Shock Files

As indicated above, these must follow the rules for GEMPACK text data files, as documented in chapter 6 of GPD-4.

In particular,

- repeated shocks of the same size can be abbreviated using the syntax

  <integer>*<value>

  For example, **20*1.3** means 20 values each equal to 1.3.

  (There should be no space before or after the '*'.) See section 6.2.7 of GPD-4.

- exponential notation (for example, 1.234E2 means 123.4) can be used – see section 6.2.6 of GPD-4.

- there is no difference between row_order and col-order for a one-dimensional array so the "how much data" part just needs a single integer followed by a semi-colon as in, for example,

  ```
  105 ;
  ```

  (if 105 components are shocked).

5.5.3 Components Must Be In Increasing Order in Most Cases

If you are only shocking some components of a variable, in most cases the components specified must be in increasing order in the shock statement on the Command file. Examples will make this clear.

Consider variable **a1prim(COM)** from ORANIG01.TAB (see section 6.3 of this document, and section 1.4.1 of GPD-8) and suppose for simplicity that COM has just 4 elements (agriculture, food, manufactures, services). Suppose also that COM has subsets COM1 = (agriculture, food) and COM2 = (services, food).

Note that the elements of COM1 are in the same order as those of COM while those in COM2 are not in the same order as in COM since services comes before food in COM2 but not in COM. We say that the elements of COM1 are in increasing order but those of COM2 are not.

The following shock statements have the components specified in increasing order.

```
Shock  a1prim 2 4 = 1.1 2.2 ;      ! increasing order
Shock  a1prim(COM1) = 1.1 2.2 ;    ! increasing order
```

The following shock statements do NOT have the components specified in increasing order.

```
Shock  a1prim 4 2 = 3.3 6.01 ;     ! not allowed (not increasing order)
Shock  a1prim(COM2) = 3.3 6.01 ;   ! not allowed (not increasing order)
```

We feel that, in these last two statements, it is at least slightly ambiguous which shock value should be associated with each shocked component. For example, does shock 3.3 go with the first component listed (component number 4) or with the first amongst the shocked components (component 2)? For this reason, these statements **are not allowed**.[43]

On the other hand, there is no ambiguity in the statements

```
Shock  a1prim 4 3 = uniform 3.3 ;  ! not increasing order but ok
Shock  a1prim(COM2) = uniform 3.3 ;   ! not increasing order but ok
```

because "uniform" indicates the same numerical shock to each component specified.

Nor does there seem to be any ambiguity in the statement

```
Shock  a1prim(COM2) = select from file a1prim.shk ;  ! allowed
```

In this case, the numbers on the file a1prim.shk are naturally associated with the components of variable a1prim.[44] Thus the order of the components in the shock statement has no effect on the meaning of the statement. For example, if COM3 = (food, services) which has elements in the same order as those in COM, this statement above is really just the same as the alternative

```
Shock  a1prim(COM3) = select from file a1prim.shk ;
```

These examples motivate the following rules which are enforced by the software.

1. If only some components are shocked, and these do not appear in increasing order in the statement, the statement is only allowed if
    - it contains the word "uniform", or

[43] It would be possible to document rules specifying how these statements would be interpreted. But, as far as possible, we want the meaning of Command file statements to be intuitively obvious, which is not the case for these statements.

[44] For example, suppose that the text file a1prim.shk is:
4 ;
12 13 14 15
Then the "select from" indicates that food (component 2 of COM) is associated with value 13 while services (component 4) is associated with value 15. [See section 5.8 below for more details about associating numbers and components on shock files.]

- the components are specified as arguments (via elements and/or subsets) and the statement includes "select from file".[45]

2. Otherwise the statement is not allowed.

If GEMSIM, a TABLO-generated program or SAGEM encounters a shock statement which is not allowed by the above rules, the error message will say that the statement is not allowed because "**the components are not in increasing order**".

We recommend that you try to make the meaning of your Command file and TABLO Input file statements as transparent as possible since others may need to read them. Hence we recommend that

- you always specify components in increasing order in shock statements.
- you always specify the elements of subsets in the same order as those in the big set.[46]

5.5.4 Additional and Target Shock (Ashock or Tshock) Statements for Dynamic Models

If you work with recursively dynamic models such as MONASH, GTAP-DYN and GTEM, you solve a base case (over several years) and then a policy run in which additional shocks are given in some of the years. [See section 10.1 of GPD-4 for a little more background on these sorts of models.]

In the policy simulations, the shocks are sometimes in addition to those given to the same variables in the base case. For example, the base case may include shocks to certain tax rates and the policy being investigated may be a further variation in these tax rates. The statement

ashock … ;

can be used to indicate that the shocks specified in it are **in addition** to any shocks to the same components of the same variable specified (in shock statements) elsewhere in the Command file.

Alternatively, the policy being investigated may specify the **total change or percentage change** in these tax rates (relative to what the rates are in the relevant year at the start of the forecast and policy runs). The statement

tshock … ;

can be used to indicate that the shocks specified overrides any shocks to the same components of the same variable specified (in shock statements) elsewhere in the Command file.[47]

These ashock and tshock statements are especially designed to be used in the policy shocks (.PSH) files used in conjunction with our Windows interface **RunDynam** (or its model-specific versions including **RunMONASH**

---

[45] You may also argue that the statement
shock a1prim 4 2 = select from file a1prim.shk ; ! not allowed
is not ambiguous, giving the same reasons that we gave when discussing the similar statement
shock a1prim(COM2) = select from file a1prim.shk ; ! allowed
We do not allow the first of these (the element numbers can be difficult to interpret if there are lots of them) but do allow the second because the use of sets and elements as arguments makes statements relatively easy to interpret. Indeed, we recommend that you avoid these issues by always putting elements of subsets in the same order as those of the big set, and always putting component numbers in increasing order.

[46] GEMPACK programs should handle subsets whose elements are not in the same order as the elements of the big set (except when this is not allowed, as, for example, in certain shock statements as indicated in the text). However it can be confusing looking at results and data when elements of a subset are not in the same order as in the big set. For example, with the sets COM and COM2 as in the text, a text data file for Coefficients COEF1 defined over COM and COEF2 defined over COM2 would be something like:
4 ; ! Values of COEF1(COM)
1 2 3 4
2 ; ! Values of COEF2(COM2)
21 22
Here the first of the COEF2 values would be associated with services (the first component in COM2) which might be confusing.

[47] "ashock" statements were new for Release 6.0, while "tshock" statements were new for Release 7.0.

and **RunGTEM**) – see section 2.7 of GPD-4. If you do not use RunDynam etc, you probably will not need to know about ashock and tshock statements.

Example

In the base case for the year 2002, there may be a shock to export taxes via the Command file statement

```
shock xtax = uniform 20 ;
```

Suppose that the policy in question involves modelling the effects of an extra 5% increase in these taxes in the year 2002. You can prepare the shocks part of the Command file for year 2002 of the policy run by transferring all shocks from the Command file for the same year of the base case (there are often lots of these statements) and then putting the statement

You can put the statement

```
ashock xtax = uniform 5 ;
```

into your Policy shock (.PSH) file. The software will combine the two shocks (20% and the additional 5%) at run time.[48] Note that this combination is done by compounding for percentage-change variables. Thus if xtax is a percentage-change variable, the shock given will be 26 per cent.[49]

Alternatively, suppose that the policy in question involves modelling the effects of a 25% increase in these taxes in year 2002 (rather than the 20% increase in the forecast run). Then the statement

```
tshock xtax = uniform 25 ;
```

could be put into the policy shocks file. [This will override the "shock xtax = uniform 20 ;" statement from the base case for this year when the shocks from the base case are appended to the extra policy shocks.]

Syntax and Semantics for ashock and tshock statements

The syntax is exactly the same as for shock statements except that the key word is "ashock" or "tshock".

"ashock" and "tshock" statements are only allowed with GEMSIM and TABLO-generated programs: they are not allowed with SAGEM.

You must not put two ashock/tshock statements for the same component of the same variable. Thus, for example, the two statements

```
ashock xtax = uniform 10 ;
ashock xtax = uniform 1 ;          !Wrong
```

would result in an error. The following two statements

```
ashock xtax = uniform 10 ;
tshock xtax = uniform 1 ;           !Wrong
```

would also result in an error.

Although ashock and tshock statements are valid for all simulations, we recommend that you only use them if you are carrying out a set of policy simulations with a dynamic model via RunDynam or one of its variants.

5.5.5 Shocking a Variable with No Components Exogenous

A statement on Command files for GEMSIM and TABLO-generated programs (but not SAGEM) is[50]

```
statements to shock variable with none exogenous are ok = yes|NO ;
```

The default is "no" which means that a statement

```
shock <variable> = … ;
```

---

[48] RunDynam prepares the shocks part of the Command file for year 2002 of the policy run by transferring all shocks from the Command file for the same year of the base case. These will include statements equivalent to "shock xtax = uniform 20 ;" from the base case.

[49] Note that (1+20/100)(1+5/100) = 1.26 so that the compounding of 20% and 5% leads to a 26% shock.

[50] This statement was introduced in Release 6.0.

is only allowed if <variable> has at least one exogenous component.

However, in handling inequalities (such as quotas) via the methods described in Bach and Pearson (1996), it is sometimes convenient to put in such a statement even though it may happen that no components of the variable in question are exogenous. [Details can be found in section 8.3 Bach and Pearson (1996).] When using the methodology described in that technical paper, you can put the statement

```
statements to shock variable with none exogenous are ok = yes ;
```

in your Command file to avoid having to comment out certain "shock" statements in those cases (not easy to predict in advance using this methodology) where no components of the variable are exogenous.

Note that the methods described in Bach and Pearson (1996) are now superseded by the methods described in chapter 16 below.

5.6 FINAL_LEVEL Statements : Alternatives to Shock Statements

In some cases the desired post-simulation values for certain variables are most naturally specified in the levels rather than as changes or percentage changes.

For example, in trade liberalization scenarios in which all trade barriers are removed, the post-simulation levels values of the relevant powers of the taxes/tariffs etc are 1. Prior to Release 8.0, modellers needed to calculate the pre-simulation levels value of the relevant power, then calculate the percentage change required to change its levels value to 1 and then enter this value as a "shock" statement.[51]

Now it is possible in many cases to specify the post-simulation levels value directly via a ***final_level*** statement.[52]

The syntax and semantics of *final_level* statements are the same as for shock statements except for the initial keyword. *Final_level* statements cannot be used with SAGEM (they are only available for GEMSIM and TABLO-generated programs).

Final_level statements can be used when the variable in question is either

1. a levels variable or the linear variable associated with a levels variable, or
2. a linear variable whose original (that is, pre-simulation) levels value has been specified via an **ORIG_LEVEL=** qualifier when the variable was declared.

In the first case the variable name after "final_level" can be either the levels variable name or the name of the associated linear variable.

> For example, in SJLB.CMF the shock statement
>
> **shock p_XFAC("labor") = 10 ; ! percentage change**
>
> could be replaced by either
>
> **final_level XFAC("labor") = 4.4 ;** ! indicates post-simulation level
>
> or
>
> **final_level p_XFAC("labor") = 4.4 ;** ! indicates post-simulation level
>
> We recommend the former since it seems more natural (and is more readily understood) to use the levels name in the context of a levels target. [Here the pre-simulation value of XFAC("labor") is 4.0 if you are using the standard data base shown in section 2.1.1 of GPD-1.]

In the Stylized Johansen example above, the levels target is less natural perhaps than the percentage change since you need to look at the starting data base to find the pre-simulation value before you can specify a sensible value for the *final_level* statement.

---

[51] The GTAP project set up the file SHOCKS.TAB and associated machinery in RunGTAP to assist modellers in computing the relevant shocks for trade liberalization scenarios.

[52] We are grateful to Yiannis Zahariadis whose question to GTAP-L about this topic suggested the desirability of introducing a final_level statement.

A second example of powers of taxes and tariffs in GTAP.TAB is instructive and more natural since there the values used in the *final_level* statement are 1 when complete removal of the tax or tariff is desired. In GTAP.TAB, **tms(i,r,s)** is a linear variable which indicates the percentage change in the power of the import tax on imports of commodity i from region r imported into region s.

The following lines could be added to the bottom of any recent GTAP.TAB (for example, GTAP61.TAB – see section 1.6.1 of GPD-8) to introduce TMS_L (power of import tariff) as a levels variable.

```
VARIABLE (Levels, Linear_var=tms)
(all,i,TRAD_COMM)(all,r,REG)(all,s,REG)      TMS_L(i,r,s)
# import tax in s on good i imported from region r (levels value) # ;

Zerodivide Default 1.0 ;
Formula (initial) (all,i,TRAD_COMM)(all,r,REG)(all,s,REG)
        TMS_L(i,r,s) = VIMS(i,r,s)/VIWS(i,r,s) ;
Zerodivide Off ;
```

[The linear variable tms is defined earlier in the TAB file. The "Linear_Var=" qualifier associates the new levels variable TMS_L with the previously declared linear variable tms – see section 2.2.2 of GPD-2.]

With this extra code in the TAB file, you can model removal of all import taxes on food from all regions [the set is REG] into the European Union (EU) via the Command file statement

final_level TMS_L("food",REG,"EU") = uniform 1 ; ! indicates post-sim levels values

instead of

shock tms("food",REG,"EU") = select from file tms.shk ; ! indicates percentage changes

The effect of the *final_level* statement is obvious and the file tms.shk is not needed. [Note the use of the levels variable name TMS_L in the *final_level* statement. The levels name TMS_L or the linear name tms could be used in either the *final_level* or shock statement. However we believe that the use of the levels name TMS_L is more natural in the *final_level* statement and that the use of the linear name tms is more natural in the shock statement.]

When you specify one or more *final_level* statements, GEMSIM or the TABLO-generated program converts the levels value into the appropriate shock value (change or percentage change). The shock values are echoed by the program immediately after the levels values are read.

In order to calculate the appropriate shock, the program must have already calculated the value of the appropriate coefficient (for example, TMS_L in the GTAP example above). For this reason, the shocks part may be carried out later in the run when *final_level* statements are present compared to what would happen if the corresponding "shock" statements were present.[53]

[53] If the only shock statement is

shock tms("food","SSA","EU") = -10 ;

then the shocks will be processed immediately after the closure and before all Reads and Fomulas. But when the statement

final_level tms("food","SSA","EU") = 1 ;

is used, the program first does all Reads and Formulas to be sure that it has calculated the values of all (user-specified) Coefficients before it processes the final_levels statement(s) (and before it processes any other "shock" statements). See section 6.2.1 for more details.

5.7 CHANGE or PERCENT_CHANGE Shock Statements

Instead of the usual shock statement there are two shock statements **percent_change** and **change** which can be used when the original levels value of a variable is known.[54]

percent_change <variable> **= .... ;**

change <variable> **= .... ;**

The syntax and semantics of *change* or *percent_change* statements are the same as for *shock* or *final_level* statements except for the initial keyword. C*hange* or *percent_change* statements cannot be used with SAGEM (they are only available for GEMSIM and TABLO-generated programs).

Change or *percent_change* statements can be used when the variable in question is either

1. a levels variable or the linear variable associated with a levels variable, or
2. a linear variable whose original (that is, pre-simulation) levels value has been specified via an (**ORIG_LEVEL= ....**) qualifier when the variable was declared.

In the first case the variable name after ***change*** or ***percent_change*** can be either the levels variable name or the name of the associated linear variable.

For example, in SJLB.CMF the shock statement

shock p_XFAC("labor") = 10 ; ! percentage change

could be replaced by either

change XFAC("labor") = 0.4 ; [55]

or

percent_change XFAC("labor") = 10 ;

Alternatively, you could use the associated linear variable p_xfac, as in

change p_XFAC("labor") = 0.4 ;

or

percent_change p_XFAC("labor") = 10 ;

However, the associated linear variable seems less natural than the levels variable (especially in the first case above).

If you are using a linear variable with an ORIG_LEVEL qualifier, you must use the linear variable name.

Example – Homogeneity Simulation

You may have a change variable d_psd_t defined in your TABLO Input file by

```
Coefficient PSD_T ;
Formula  PSD_T = .... ;
Variable(change, orig_level = PSD_T)  d_psd_t ;
```

Here PSD_T is the value of some tax and the variable d_psd_t is the change in the value of this tax.[56]

Suppose that you are performing a homogeneity simulation (see section 13.2 of GPD-4) in which all nominals should be increased by one percent.

If the value of PSD_T calculated via the formula is

---

[54] Change and Percent_Change statements were introduced in Release 8.0.

[55] The pre-simulation levels value of XFAC("labor") is 4.0 if you are using the standard data base shown in section 2.1.1 of GPD-1.

[56] This example is taken from the MONASH model, Dixon and Rimmer (2002).

PSD_T=162330

then a one percent shock for this homogeneity test would have to be calculated outside the model to be 1623.30 and the shock statement in your Command file would be

```
shock d_psd_t = 1623.30;
```

This shock statement can be replaced by the statement

```
percent_change  d_psd_t = 1.0 ;
```

This statement is still correct even if the data read in changes the value of PSD_T from 162330 to some other value. You do not need to recalculate the one percent change in PSD_T, or change the Command file *percent_change* statement.

5.8 Shock Files – Fine Print

Shock files were introduced in section 5.5.2 above. That section contains several examples.

In most cases, it will be clear how the numbers on the file are associated with the shocked components of the variable.

In this section we document this association in complete detail. We also describe the checks made on the size of the array on the file and the shocked components. We give examples to illustrate the alternatives.

5.8.1 All Components Shocked and "Select From File" Cases

This section covers the following two cases:

1. Any shock statement containing the words "select from file"

   ```
   Shock x [components] = select from file <file_name> [header "<header>"];
   ```

   whether or not there are any components specified in the statement, and irrespective of whether or not all components of the variable are shocked.

2. Any shock statement using the word "file" but not the words "select from file" in which all components of the variable are exogenous and shocked. That is, a statement of the form

   ```
   Shock x = file <file_name> [header "<header>" ] ;
   ```

   when all components of variable **x** are exogenous.[57]

Section 5.8.2 below covers the case where only some components are shocked and "select from file" is not in the statement.

As explained in section 5.5.2, if the header part if present, the file must be a Header Array file, otherwise it must be a GEMPACK text data file.

Case 1 – Dimensions Must Match if Array on File is Two or Higher Dimensional

- The dimensions of the array on the file must match those of the variable **x** in the same way as in a READ statement from this part of the file into a Coefficient with the same arguments as variable **x**.[58]

---

[57] If only some components of variable **x** are exogenous, then only these exogenous components are shocked (see section 5.11.1) and then section 5.8.2 below applies.

[58] This checking was introduced in Release 8.0. Prior to that, the only checking was that the size of the array on the file is equal to the number of values expected from the shock statement. For example, if the variable **x** has dimensions COMxIND and if COM has 10 elements and IND has 12 elements then arrays on the file of any of the sizes 60x2, 3x40 or 10x12 were allowed prior to Release 8.0. For Release 8.0 or later, only arrays of size 10x12 (or 10x12x1) or 120 (see Case 2 in the text below) are allowed. We have excluded the 60x2 cases and 3x40 cases since it is not at all obvious how to associate the data from the array on the file to the different components of the variable being shocked.

- The assignment of values on the file to components of variable **x** is done in the same way as the assignment for a READ.

[See sections 4.10.1 and 4.10.2 of GPD-2 for documentation for reads into a Coefficient.]

Examples will make this clear.

Consider variable **a3(COM,SRC)** from ORANIG01.TAB (see section 6.3 of this document and section 1.4.1 of GPD-8) and suppose for simplicity that COM has just 3 elements (agriculture, manufactures, services) and that SRC has the usual elements (dom, imp). Note that, as indicated in section 5.3, the order of the components of variable a3 is as follows:

a3("agriculture","dom")	component number 1
a3("manufactures","dom")	component number 2
a3("services","dom")	component number 3
a3("agriculture","imp")	component number 4
a3("manufactures","imp")	component number 5
a3("services","imp")	component number 6

Example 1. Consider the statement

```
Shock a3 = file a3.shk ;
```

In this two or higher dimensional case, the text data file a3.shk must contain a 3x2 array on the file (or extra 1's are allowed so that 3x1x2 and 3x2x1x1 etc would be allowed). [A one-dimensional array is also allowed – see Case 2 below.]

Suppose that the array on the file is

```
3  2 ;
1.0  2.0
3.0  4.0
5.0  6.0
```

This is in (the default) row_order (see sections 6.1 and 6.2.3 of GPD-4). Then the shock to a3("manufactures","imp") will be 4.0 since 4.0 is in the ("manufactures","imp") position of the array on the file. [The ("manufactures",SRC) part of the array on the file is the second row in that array, which corresponds to the second line of data on the file.]

Suppose, on the other hand, the array on the file is in col_order (see sections 6.1 and 6.2.4 of GPD-4):

```
3  2  col_order ;
7.0    8.0    9.0
10.0  11.0  12.0
```

Then the shock to a3("manufactures","imp") will be 11.0 since 11.0 is in the ("manufactures","imp") position of the array on the file. [The (COM,"imp") part of the array on the file is the second column in that array, which corresponds to the second line of data on the file.]

Example 2. Consider the statement

```
Shock a3(COM,"imp") = select from file a3.shk ;
```

Again, in this case two or higher dimensional, the text data file a3.shk must contain a 3x2 array on the file (or extra 1's are allowed so that 3x1x2 and 3x2x1x1 etc would be allowed).

Suppose that the array on the file is

```
3  2 ;
21.0  22.0
23.0  24.0
25.0  26.0
```

Then the shock to a3("services","imp") will be 26.0 since 26.0 is in the ("services","imp") position of the array on the file.

Suppose, on the other hand, the array on the file is in col_order:

3 2 col_order ;
37.0 38.0 39.0
40.0 41.0 42.0

Then the shock to a3("agriculture","imp") will be 40.0 since 40.0 is in the ("agriculture","imp") position of the array on the file.

Example 3. Consider the statement

```
Shock a3 2 5 6 = select from file a3.shk ;
```

Again, in this two or higher dimensional case, the text data file a3.shk must contain a 3x2 array on the file (or extra 1's are allowed so that 3x1x2 and 3x2x1x1 etc would be allowed).

Suppose that the array on the file is

3 2 ;
51.0 52.0
53.0 54.0
55.0 56.0

Thus the shocked components are a3("manufactures","dom") [component number 2 – see above], a3("manufactures",imp") [component number 5] and a3("services","imp") [component number 6].

The shocks (which can be read off from the appropriate part of the array on the file, concentrating on the arguments in the paragraph above, not the component numbers) will be

- 53.0 to a3("manufactures","dom") [the ("manufactures",SRC) part of the array on the file is the second line of data on the file],
- 54.0 to a3("manufactures","imp"), and
- 56.0 to a3("services","imp").

Case 2 – One-Dimensional Array on the File

❖ When the variable **x** being shocked has a single argument or index [for example, x(i) for i in IND], it is natural and apropriate to have a one-dimensional array on the file.

❖ When the variable **x** being shocked has two or more arguments [for example, x(c,i) for c in COM and i in IND], it is more natural to have an array with the same number of dimensions on the file (as in Case 1 above). However a one-dimensional array on the file is allowed for backwards compatibility with older versions of GEMPACK. For new shock files, we recommend you follow the rules in Case 1 above.

Suppose that the variable **x** has N components in total. Then

- the array on the file can be a one-dimensional array (that is, a vector) of size N.
- the assignment of numbers on the file to components of **x** is done using the component numbers of **x** as documented in section 5.3.

Again examples will make this clear. In these example, consider again variable **a3(COM,SRC)** from ORANIG01.TAB.

Example 1. Consider the statement

```
Shock a3 = file a3.shk ;
```

The text data file a3.shk could contain a one-dimensional array of size 6 on the file (or extra 1's are allowed so that 6x1 and 1x1x6 etc would be allowed). [A 2-dimensional array on the file is also allowed – see Case 1 above.]

Suppose that the array on the file is

6 ;
11.0 12.0 13.0 14.0 15.0 16.0

Then the shock to a3("manufactures","imp") will be 15.0 since ("manufactures","imp") is the fifth component of variable a3 (see above) and 15.0 is in fifth number on the file.

Example 2. Consider the statement

```
Shock a3(COM,"imp") = select from file a3.shk ;
```

Suppose that the array on the file is

6 ;
21.0 22.0 23.0 24.0 25.0 26.0

Then the shock to a3("manufactures","imp") will be 25.0 since ("manufactures","imp") is the fifth component of variable a3 (see above) and 25.0 is the fifth number on the file.

Example 3. Consider the statement

```
Shock a3 2 5 6 = select from file a3.shk ;
```

Suppose that the array on the file is

6 ;
51.0 52.0 53.0 54.0 55.0 56.0

Then the shocks will be

- 52.0 to a3("manufactures","dom") [component number 2 – see above],
- 55.0 to a3("manufactures","imp") [component number 5], and
- 56.0 to a3("services","imp") [component number 6].

5.8.2 Shocking Some Components (Not "Select From File")

This section covers the case

Shock x [components] = file <file_name> [header "<header>"] ;

where **not all components of variable x are shocked** and "select from file" is not in the statement.[59]

Suppose that variable **x** has N components in total and that K components (K < N) are shocked in this statement.

In this case,

- the dimensions of the array on the file are allowed to be any which give a total of K numbers on the array.
- the assignment of numbers on the file to shocked components of variable **x** is done using component numbers in the array (as in section 5.3). The first shocked component of **x** is given a shock equal to the first component of the array on the file. The second shocked component of **x** is given a shock equal to the second component of the array on the file, and so on.

Again examples will make this clear. In these example, consider again variable **a3(COM,SRC)** from ORANIG01.TAB.

Example 1. Consider the statement

```
Shock a3(COM,"imp") = file a3a.shk ;
```

There must be exactly 3 numbers on the file.

Suppose that the array on the file is

3 ;
31.0 32.0 33.0

Then the shock to a3("manufactures","imp") will be 32.0 since ("manufactures","imp") is the second amongst the a3(c,"imp") [c in COM] components of a3 (see above) and 32.0 is in second number on the file.

[59] This includes the case "shock x = file … ;" where no components are specified in the shock statement but only some components of variable **x** are exogenous, since then only these exogenous components are shocked (see section 5.11.1). The "select from file" case is covered in section 5.8.1 above.

Example 2. Consider the statement

```
Shock a3 1 2 5 6 = file a3b.shk ;
```

There must be exactly 4 numbers on the file.

Suppose that the array on the file is

4 ;
41.0 42.0 43.0 44.0

Then the shock to a3("manufactures","imp") [component number 5 of a3 (see above), and the third component in the list "1 2 5 6" of component numbers in the "shock" statement] will be 43.0 since 43.0 is the third number on the file.

Suppose, on the other hand, that the array on the file is

2 2 ; ! row_order (see section 6.1 of GPD-4) **Bad Example**
51.0 52.0
53.0 54.0

The array is read in assuming row_order (value 53.0 is in row 2 and column 1) and is stored on the computer in the order of the component numbers for a 2x2 matrix (as specified in section 5.3[60]):

51.0 53.0 52.0 54.0

Then the shock to a3("manufactures","imp") [component number 5 of a3 (see above), and the third component in the list "1 2 5 6" of component numbers in the "shock" statement] will be 52 since 52 is in the component number 3 place (that is, row 1 and column 2) in the 2x2 array on the file (see section 5.3).

We think that the latter example in Example 2 above (2x2 matrix on the file) is particularly difficult to understand (it took us a long time to understand) and **recommend that you never specify shocks in such a complicated way**.

[60] Think of the matrix entries as **a(i,j)** where i is the row number and j is the column number. Then the first index "i" varies faster, as indicated in section 5.3. Thus a(2,1)=53.0 is the second "component" in the matrix.

5.9 Checking the Closure and Shocks

If a Command file contains statements specifying the closure and shocks, TABLO-generated programs and GEMSIM can check the closure and read in the shocks even if you don't want the simulation run.

An example is the following Command file for Stylized Johansen.

```
Auxiliary files = sj ;
file iodata = sj.dat ;
exogenous  p_XFAC ;
rest endogenous ;
shock p_XFAC("labor") = file lab.shk ;
neq = yes ;
```

Because of the "neq = yes ;" statement (see section 6.1.7), the equations will not be calculated numerically so no simulation will be carried out. However, because of the closure-related and shock statements, GEMSIM or the TABLO-generated program will set up the closure (and report if it seems not to be a valid closure) and read the shock values (from the file lab.shk in the example above). In particular, this will catch any errors in the Command file or any shocks files referred to in the Command file.

With a more complicated model, such Command files can be a useful way of checking that the closure and shocks statements are valid. For example, if you plan to set several long simulations running overnight, you may like to use the method above to get early warning of any errors in the closure and shocks part. This will also tell you if any of the required files (Environment or shocks files, for example) are missing.

To carry out such checks, all you need to do is to add the statements

neq = yes ; ! No equations, hence no simulation
nwr = yes ; ! No writes
nds = yes ; ! No displays
assertions = no ; ! No assertions

to the Command file. These tell the program to do none of the usual actions (that is, no equations and hence no simulation, no writes, no displays and no assertions – see section 6.1.7).[61] [Of course you must comment these out when you really want the simulation to run.]

The closure check carried out in these cases is just the initial check (see section 5.2.7) that the number of endogenous variables equals the number of equations. No check is made that the LHS Matrix is non-singular.

Note that, even with a large model for which a simulation may take a long time, the program will take little time (usually less than a minute) to check the closure and shocks.

If there are no shock statements, the program will still check the closure.

If you use RunDynam (or model-specific variants such as RunMONASH) – see section 2.7 of GPD-4 – you can ask for all closures and shocks to be checked via the **Tasks** menu.

[61] Under Release 5.2 of GEMPACK, the program would have told you that there are no actions, hence nothing to do. Release 6.0 (and later) programs regard checking the closure and shocks as actions worth doing.

5.10 Levels Variable Names can be Used in Command Files

You can use levels names for variables in Command files for GEMSIM and TABLO-generated programs instead of the associated linear variable names.[62]

For example, in SJLB.CMF you could write

exogenous XFAC ; ! instead of "exogenous p_XFAC ;"

shock XFAC("labor") = 10 ; ! instead of "shock p_XFAC("labor") = 10 ;"

You can use levels variable names with GEMSIM and TABLO-generated programs when you are specifying the closure, the shocks, subtotals, cumulatively-retained endogenous and XAC-retained variables.

You cannot use levels variable names in Command files for SAGEM.

5.11 Clarification Concerning Shocks on a Command File

This section aims to clarify issues concerning shocks on a Command file which were ambiguous in the Release 6.0 and earlier documentation.[63]

5.11.1 Shock Statement When Only Some Components are Exogenous

Consider the statement

```
shock <variable-name> = … ;
```

when only some components of <variable-name> are exogenous. The above statement (which has no component numbers or arguments after the variable name) means that ALL EXOGENOUS components of the variable are to be shocked. The information after the '=' sign must be consistent with this (correct number of shocks etc).

In particular, consider the statement

```
shock <variable-name> = select from file … ;
```

when only some components of the variable are exogenous. In this case, the file must contain N values (if N is the number of components of the variable) and shocks will be given to all the exogenous components of the variable. [As usual with a "select from" statement (see section 5.5.2), the shocks for the exogenous components of the variable will be selected appropriately from amongst the N values on the file.]

5.11.2 No "Shock" Statements on a Command File

If a Command file contains no "shock …" statements, the default with Release 5.1 was to give shocks of one to all exogenous variables (see the "SHOCK RELATED" part of section 18.1). We believe that this was a dangerous default since it would apply if you inadvertently left out all shock statements. Accordingly we changed this with Release 5.2. Now **you must have at least one "shock …" statement in every Command file** for GEMSIM or a TABLO-generated program or SAGEM. If there are no shock statements, the program will stop with an error.

---

[62] This was introduced in Release 8.0.

[63] It also serves to note a change in the situation which applied in Release 5.1 when no shock statements are included on a Command file.

5.12 Specifying the Shocks Interactively

Since Command files are always available when you must specify shocks, we strongly recommend that you always use them since specifying shocks interactively (or even on a Stored-input file) is less reliable and less transparent. Accordingly **you may prefer to skip this section**.

The steps involved in specifying shocks interactively are roughly as follows. Prompts are given at all steps. You are first asked

Are all shocks are equal to 1? [y/n]

If you respond 'y' (yes), this completes the specification of the shocks.

If you respond 'n', you are presented with a list of the relevant variables (namely, those with at least one component shocked) in turn, one at a time. They are presented in the order in which the variables are declared in the TABLO Input file for the model.

For a variable with only one component shocked, you are asked to enter the numerical shock.

For a variable with more than one component shocked, the choice is a little more complicated.

You are first asked if you want to give **uniform** shocks (that is, the same shock to each shocked component – see section 5.5). If you respond 'y' (yes), you are next asked to enter the value of this uniform shock.

If you respond 'n' (not uniform shocks), you are asked whether you want to read the shocks from a file (a text file) or the terminal. In the case of a file, you are then asked for the name of the file, after which the program reads the values from that file; see section 5.5.2 for advice about preparing these files. In the case of the terminal, you are then asked to enter the appropriate number of shock values.

This is repeated for each variable with at least one component shocked.

CHAPTER 6

6. Actions in GEMSIM and TABLO-generated Programs

6.1 Possible Actions in GEMSIM and TABLO-generated Programs

When you run GEMSIM or a TABLO-generated program and attach the data, the program can carry out various **actions** (see Figure 6.1).

6.1.1 Multi-Step Simulations with Economic Models

TABLO Input files for economics models are those containing EQUATION statements. The most important action for the associated TABLO-generated program is that of carrying out a (possibly multi-step) simulation with the model (the first case illustrated in Figure 6.1). This produces a Solution file containing simulation results (percent changes and possibly pre- and post-simulation levels results). It also produces updated data files and other potentially useful files (including Equations and Environment files).

Models with No Update Statements

It should be possible to carry out multi-step simulations with all economic models. This is because the underlying levels equations of an economic model are bound to be nonlinear so that multi-step calculations are required to solve them accurately.

However it only makes sense to carry out a multi-step simulation with models whose TABLO Input files say how the data is to be updated after each step of the calculation (see sections 2.13.3 and 3.4.4 of GPD-1). This is why it is not possible to carry out a simulation with TABLO-generated programs whose TABLO Input files contain only linearized EQUATIONs but contain no UPDATE statements or levels VARIABLEs. For these models, the most important action is that of creating the Equations file for the model (see the next section). It is only possible to carry out a Johansen simulation with these models. That must be done by using SAGEM, starting from the Equations file.[64]

6.1.2 Creating an Equations File

If a TABLO Input file contains equations, GEMSIM or the associated TABLO-generated program can be used to create an Equations file. [This is the second case illustrated in Figure 6.1.] You may want to create the Equations file in order to investigate closure or homogeneity problems using SUMEQ (see chapter 13 of GPD-4). This Equations file can also be the starting point from which you can carry out Johansen simulations by running the program SAGEM.

[64] The only example amongst the models usually supplied with GEMPACK is the DMR model (see section 1.8 of GPD-1). Of course, this could have UPDATE statements added to it, as explained in Chapter 4 of DPPW.

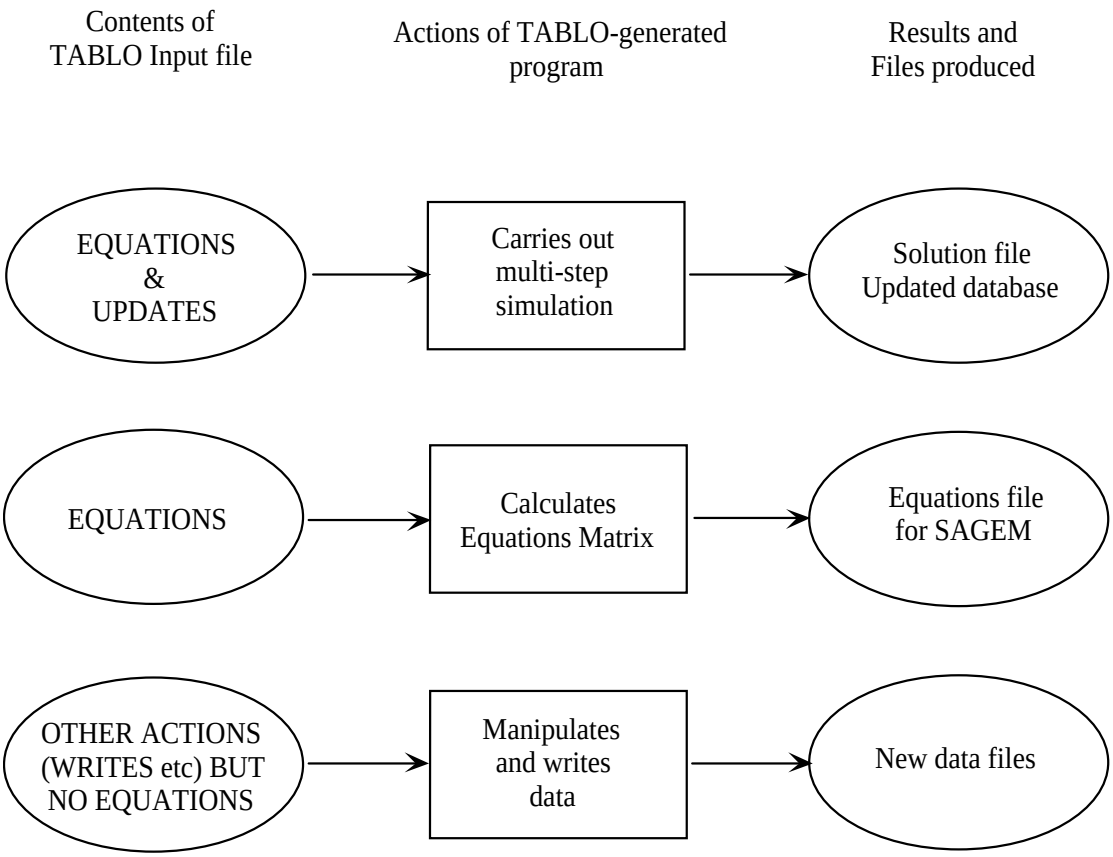


Figure 6.1: Actions of GEMSIM and TABLO-generated Programs

6.1.3 Other Actions (Writes, Displays, Assertions, Range Tests and Transfers)

Other possible actions are those of doing the

- DISPLAYs (if there are any) in your TABLO Input file – see section 4.3.
- WRITEs (if there are any) in your TABLO Input file – see section 4.1.1.
- ASSERTIONs (if there are any) in your TABLO Input file – see section 6.3.
- TRANSFERs (if there are any) in your TABLO Input file – see section 6.5.
- Range tests (if there are any) in your TABLO Input file – see section 6.4.

These other actions may include "**extra**" TABLO-like statements on the Command file, including xwrite, xdisplay and xtransfer statements (see section 6.6).

6.1.4 Data Manipulation

Sometimes TABLO Input files are created for data manipulation rather than for modelling (see section 4.4 of GPD-1 for an example). In this case there are no EQUATIONs (and hence no VARIABLEs or UPDATE statements) and the only possible actions are the Other Actions (WRITEs, DISPLAYs, ASSERTIONs, TRANSFERs or Range tests). [This is the third case illustrated in Figure 6.1 .] The WRITEs are usually the main actions since they can produce new (or modified) data files.

If you wish to look at the values of Coefficients and expressions in your data-manipulation TAB file, you can do so via AnalyseGE if you produce a CVL file (see section 8.6).

6.1.5 "Extra" (TABLO-like) Actions

Note that TABLO-generated programs and GEMSIM are also usually able to carry out some "extra" actions (TABLO-like statements in the Command file rather than in the TABLO Input file) – see section 6.6. Usually these "extra" actions are carried out during the reads/formulas part of step 1 (see Figure 6.2).

6.1.6 Checking the Closure and Shocks

Another possible action is described in section 5.9, namely that of checking the closure and shocks (without doing a simulation). Section 6.2.1 below tells you when this checking is done.

6.1.7 Controlling Whether and How the Actions are Carried Out

Unless you directed otherwise (via the options menu at the start of the CODE stage of TABLO – see section 5.1.1 of GPD-2), GEMSIM and/or the TABLO-generated program are capable of carrying out all the actions in the TABLO Input file. When you run either GEMSIM or the TABLO-generated program, it will normally carry out all of these actions.

However you can control whether and how these are carried out. The Command file statements[65]

```
neq = yes ;  ! don't do equations
nwr = yes ;  ! do no writes
nds = yes ;   ! do no displays
Assertions = yes|no|warn ; ! see section 6.3
Range test initial values = yes|no|warn ; ! see section 6.4.4
Range test updated values = updated|extrapolated|both|no|warn ;  ! section 6.4.4
```

can be used to stop writes, displays or control how assertions and range tests are carried out.

If you include the statement

```
simulation = no ;  ! don't do simulation
```

then the equations can be calculated (and the Equations file written) but no simulation is carried out. If you start from an existing Equations file and include this statement, you can still set up and save a closure on an Environment file.

Of course, equations are essential for a simulation and so, if you include "neq = yes ;" (Do no equations), this means that no simulation will be carried out either, and the only actions possible are the "Other actions" (writes etc).

If you include the statement

```
nud = yes ;  ! don't do final updates
```

the **final** updated data files are not written (but intermediate ones – see section 6.2 – are calculated since this is essential for calculating a multi-step solution).

6.1.8 Some READs and FORMULAS May be Omitted

If you tell the program not to carry out some actions it is capable of (for example, by including the statement "nds = yes ;" in your Command file when the TABLO Input file contains DISPLAYs), the program may not do all READs and FORMULAs in the TABLO Input file. It only does enough to calculate the values of all

---

[65] Alternatively you can use the options menu (see chapter 14) in a similar way. [For example, selecting option NSM is the same as putting "simulation = no ;" in your Command file.] But not all of the Command file statements shown have option equivalents. We recommend using Command file statements rather than options since then these become part of the record of the run (in the Command file).

COEFFICIENTs required to carry out the actions requested. For example, consider the following simple TABLO Input file.

```
COEFFICIENT C1 ; C2 ; C3 ;
READ C1 FROM TERMINAL ;
FORMULA C2 = C1 + 1 ;
FORMULA C3 = C1 + 2 ;
DISPLAY C2 ;
WRITE C3 TO TERMINAL ;
```

If you include "nds = yes ;" in your Command file, the FORMULA for C2 will not be calculated (since the value of C2 only needs to be calculated in order to display it). But if the FORMULA for C3 were changed to

```
FORMULA C3 = C2 + 1 ;
```

then the FORMULA for C2 would be executed since its result is required to calculate C3.

This principle is also applied to READs and FORMULAs at steps 2,3… of a multi-step calculation. If the values of some COEFFICIENTs are only needed to calculate entries of submatrices involving a variable which is exogenous and not shocked, the READs and/or FORMULAs for these COEFFICIENTs will not be carried out at steps 2,3… (but they will be at step 1 since the closure is usually not known at that time).

6.1.9 Writes and Displays at All Steps of a Multi-step Simulation

Normally WRITEs and DISPLAYs are only done during step 1 of a multi-step simulation (when they reflect values on the original data files). Occasionally, perhaps because you are trying to identify a problem with your UPDATE statements, you may wish to see these values at all steps of the simulation. If you include the statement[66]

`dws = yes ;` ! Displays and Writes to text files at all Steps

in your Command file, all displays and writes to the terminal (but not to Header Array files or other text files) will be done at each step.

6.1.10 Echoing Activity

During a run of GEMSIM or a TABLO-generated program, "echoing activity" means indicating all reads, formulas, set, subsets, displays, writes, backsolves and updates as they are carried out, and reporting how many nonzeros are in each submatrix as it is calculated.

If you are carrying out a simulation, the default is to echo activity during the first step only. In this case, if you include the statement[67]

`eaa = yes ;` ! Echo All Activity

activity is echoed during all steps of a multi-step simulation.

If you are not carrying out a simulation, the default is to echo all activity. In this case, if you include the statement[68]

`eaa = no ;` ! do not Echo All Activity

activity is not echoed.

6.1.11 Writes to the Terminal

By default WRITEs to the terminal are done in row order. If you include the statement[69]

`twc = yes ;` ! Terminal Writes in Column order

---

[66] Alternatively, you can select option DWS (see section 14.5).

[67] Alternatively, you can select option EAA (see section 14.5).

[68] Alternatively, you can de-select option EAA [that is, respond **-EAA**] (see section 14.5).

[69] Alternatively, you can select option TWC (see section 14.5).

in your Command file, all terminal writes will produce arrays written in column order (that is, col_order) as described in chapter 6 of GPD-4.

6.2 How These Programs Carry Out Multi-step Simulations

Each step of a single multi-step simulation consists of five main parts:

1. sets, subsets, reads, writes, displays, assertions, transfers, calculations of formulas, and any "extra" statements,[70]
2. calculation of equations, one submatrix (see chapter 13 of GPD-4) at a time,
3. the simulation part (solving the equations),
4. backsolving (if required),
5. updating the data.

(See Figure 6.2 for these parts, some of which are shown there in a little more detail.)

Note that, in each step, all reads and formulas are carried out before any equations are processed. This means that the values of Coefficients in the Equations are those after ALL formulas and reads, not just after all formulas and reads which occur in the TABLO Input file before the equation. See section 4.15.6 of GPD-2 for more details.

For a Johansen simulation, each of the above is done just once. For a single N-step simulation, each is done N times if you are using Euler's method or the midpoint method while each is done (N+1) times if you are using Gragg's method.[71]

If you are extrapolating on the basis of two or more multi-step solutions, each solution is calculated (first the one with the smaller number of steps, then the one with the next most, and so on).

Finally, the extrapolation is carried out. This involves taking an appropriate linear combination of the different solutions (and updated data bases).

Normally DISPLAYs, WRITEs, TRANSFERs and any "extra" actions are only done during part 1 above of the first step of the first multi-step solution, when values written or displayed reflect values in the original data. [If you are using existing Equations and Base Coefficient Values files, or an LU file, to start the simulation (see sections 9.2.1 and 9.3), note that 1 and 2 above are not done during step 1, so that DISPLAYs, WRITEs, TRANSFERs and "extra" actions are usually not done.]

An ASSERTION may be done in each step or just in the first step, depending on the qualifier (if any) in the ASSERTION statement in the TABLO Input file. [The qualifier ALWAYS, which is the default, means that the assertion is tested at each step, whereas the qualifier INITIAL means that the assertion is only tested in the first step. See section 3.14 of GPD-2.]

[70] Processing the closure and shocks (see section 6.2.1) is done here during the first step only.

[71] The output from GEMSIM and TABLO-generated programs distinguishes between "steps" and "passes" (see section 12.2). Thus a 6-step Gragg does 7 passes.

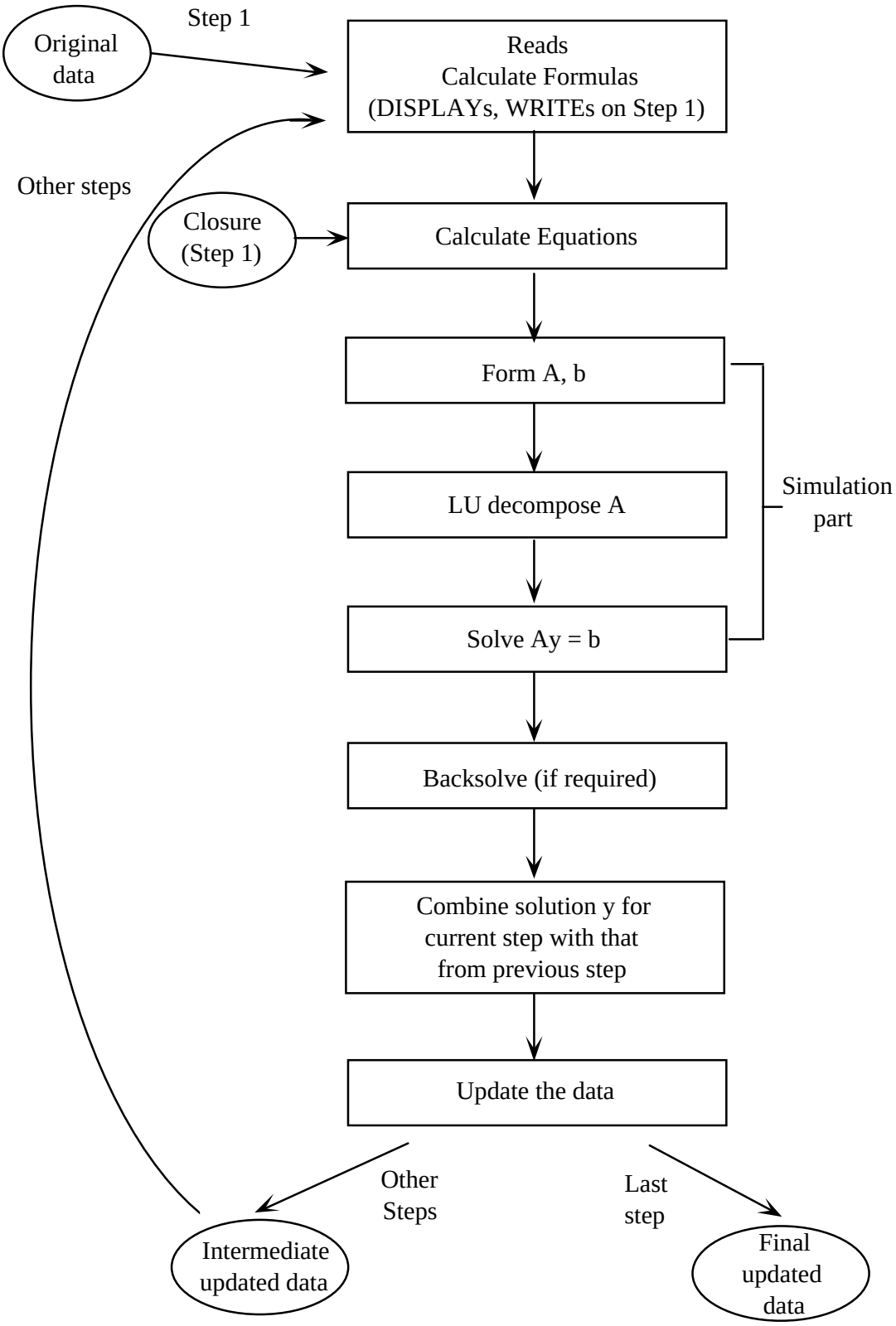


Figure 6.2: Calculation of One Multi-step Solution

In an N-step simulation, the shocks are broken into N equal parts. In step number K, the formulas and equations are calculated on the basis of the data as updated after step K-1 (if K is at least 2, or the original data if K=1), and the effect of applying just the part of the shock relevant to this step is calculated; this gives a solution, say y(K) for just these shocks. (See below for more explanation as to how y(K) is calculated.) This solution y(K) is added to the combined solution x(K-1) before this step to produce the combined solution x(K) after K steps. [x(K) reflects the movements in the endogenous variables corresponding to just K Nths of the total shocks.] Thus,

```
x(1) = y(1)      step 1
x(2) is obtained from x(1) and y(2)
x(3) is obtained from x(2) and y(3), and so on.
```

We give a concrete example in section 6.2.2 below.

At the end of each step the data is updated on the basis of the step solution y(K). Specifically, the solutions in y(K) are applied to the updated data after step K-1 to produce the updated data after step K. This data is the starting point for step K+1 (or is the final updated data at the end of this multi-step simulation if this is the last step).

At step K, the linear equations (based on the data at the start of this step) are formed up as

```
C(K).z = 0
```

much as in equation (1) of section 2.13.1 of GPD-1. Note that, in a multi-step simulation, the Equations Matrix C changes from step to step reflecting the changes in the data as it is updated, which is why we use the notation C(K) above. Then, taking account of the closure and of the shocks for this step, the equations are expressed as

```
A(K).y(K) = b(K)
```

much as in equation (3) of section 2.13.2 of GPD-1. Here the LHS matrix A(K) and the RHS vector b(K) depend on K as does the solution y(K) for this current step. These equations are solved as described in section 12.1 below: first the LU decomposition of A(K) is carried out (this is the time-consuming part) and then the values of y(K) are calculated. The solution y(K) is normally made as accurate as possible by using iterative refinement (see section 12.1 below).

Note that, if you condensed your model, some backsolving may be done even if you didn't explicitly ask for the values of any eliminated (endogenous) variables to be calculated (see section 2.3.2 of GPD-2). These are done if they appear to speed up the updates (which are done after backsolving).

6.2.1 Processing the Closure and Shocks in GEMSIM or TABLO-generated Programs

GEMSIM and TABLO-generated programs aim to process the closure (the initial check – see section 5.2.7) and the shocks as early as possible.

Often

- they process the closure as soon as all SET and SUBSET statements (including any "extra" ones – see section 6.6) have been processed (this usually happens on the preliminary pass – see section 13.2).
- they process the shocks as soon as all SET and SUBSET statements and all WRITEs to text files or to the terminal (including any "extra" ones) have been processed.[72]

If the closure or shocks depend on the value of any Coefficients[73], this processing is delayed until the values of these Coefficients are known.

The closure and shocks are processed as early as possible to give you quick feedback on any problems with them. The shocks are delayed until after any write to text files so that it is possible to write a shocks file and use it on the same run.

---

[72] This is earlier than they processed the closure and shocks in Release 5.1.

[73] Sets can depend on data (see section 4.6.5 of GPD-2). Shocks determined via *Final_level*, *Change* or *Percent_change* statements (see sections 5.6 and 5.7) cannot be processed until the values of the relevant Coefficients are known.

6.2.2 Results of a 4-step Simulation Looked at in Detail

In this section we look at the results of the 4-step version of the simulation carried out in section 2.4 or 2.6 of GPD-1. This is the simulation with Stylized Johansen in which the supply of labor is increased by 10%. We show how the shock for each of the steps is calculated, and show how the final results for certain endogenous variables are built up from the results of the individual steps.

Note that it is not necessary to understand this in order to carry out multi-step simulations with GEMPACK, so you may **prefer to skip this section**.

The results discussed in this section are those obtained using Euler's method. (Gragg's method or the midpoint method would give different results.)

(a) How the Shock is Broken Up

Imagine that the initial supply of labor is 100 units; we show how the 10% shock to it is broken up into the shock for each of the 4 steps.[74]

Over the whole simulation, the labor supply must increase by 10%, that is, from 100 to 110, which is an overall increase of 10 units. The shocks at each step are always calculated to ensure that

the same increase (in levels) occurs at each step.

Thus, at each step, the supply must increase by 2.5 (=10/4) units.

In step 1, supply must increase from 100 to 102.5 which is an increase of (25/100)x100=2.5%.
In step 2, supply must increase from 102.5 (its value after step 1) by a further 2.5 units to 105 units; this is an increase of (25/102.5)x100=2.439%.
Similarly in step 3 the increase is (2.5/105)x100=2.381%
while in step 4 it is (2.5/107.5)x100=2.326%.

Step	Supply at start	Increase	Supply at end	% Increase
1	100	2.5	102.5	2.5
2	102.5	2.5	105	2.439
3	105	2.5	107.5	2.381
4	107.5	2.5	110	2.326

(b) Calculation of Results for Endogenous Variables

Consider first what happens to just one of the endogenous variables, namely p_XH("s1"), as these four partial shocks are applied to the model.

In the first step, the levels variable XH("s1") increases by 1.5%. For simplicity (again this makes no difference to the final outcome) we suppose that the initial value of the levels variable XH("s1") = 100 as well. After the first step, XH("s1") = 101.5 units.[75]

In the second step, in response to the 2.439% shock to XFAC("labor"), the increase in XH("s1") = 1.463%, the updated value of the levels variable XH("s1") is calculated from

XH("s1") = 101.5 * (1 + 1.463/100) = 102.985 units

an increase in the combined solution of 2.985%.

---

[74] The assumption about the initial supply being 100 is, in fact, irrelevant here. You can easily check this by replacing 100 by some other number (perhaps 4, which would be the supply for the data in Table 2.1.1a of GPD-1 if the price is one dollar per unit).

[75] Note that, as you would expect, in the first step, the 1.5% increase in XH("s1") is exactly one quarter of what it would be in the corresponding 1-step simulation (namely 6.0%, as you can see by looking at the 1-step column in the Extrapolation Accuracy file SJLB.XAC discussed in section 2.13.3 of GPD-1) since the shock is exactly one quarter.

In the third step, in response to the 2.381% shock to XFAC("labor"), the increase in XH("s1") = 1.429%, and so the levels variable XH("s1") is updated to

XH("s1") = 102.985 * (1 + 1.429/100) =104.457

In the fourth step, in response to the 2.381% shock to XFAC("labor"), the increase in XH("s1") = 1.395%, and so the levels variable XH("s1") is updated to

XH("s1") = 104.457 * (1 + 1.395/100) =105.914

an overall increase in the four steps of 5.914%

Using the notation from section 6.2 above, namely letting y(K) denote the step solution at the Kth step, and x(K) the combined solution after K steps, the results for p_XH("s1") are summarised in the following table.

Step	y(K)	x(K)	
1	1.5	1.5	x(1) = y(1)
2	1.463	2.985	x(2) calculated from x(1) and y(2)
3	1.429	4.457	x(3) calculated from x(2) and y(3)
4	1.395	5.914	x(4) calculated from x(3) and y(4)

Thus the 4-step simulation result for the variable p_XH("s1") [in response to the 10% shock to XFAC("labor")] is 5.914%. This is the result for the 4-step Euler solution (not the extrapolated solution) on the Extrapolation Accuracy file SJLB.XAC. [The results are shown in section 2.13.3 of GPD-1.]

The table below shows the same information for the endogenous variable p_XH("s2").

Step	y(K)	x(K)	
1	1.75	1.75	x(1) = y(1)
2	1.707	3.487	x(2) calculated from x(1) and y(2)
3	1.667	5.212	x(3) calculated from x(2) and y(3)
4	1.628	6.925	x(4) calculated from x(3) and y(4)

Thus the 4-step Euler simulation for the variable p_XH("s2") [in response to the 10% shock to XFAC("labor")] is 6.925%.

(c) Updates and Values of Coefficients

See sections 3.4.5 to 3.4.7 of GPD-1 for numerical details about the effect of Update statements, the values of Coefficients from step to step and the values of the coefficients of the linearized equations from step to step in this 4-step simulation. [In those sections of GPD-1, the simulation is based on SJLN.TAB instead of SJ.TAB but the results are essentially identical in these two cases.]

6.2.3 Are the Linear Equations in the TAB File Satisfied by the Accurate Results?

Consider the simulation with Stylized Johansen from chapter 2 of GPD-1. This is the simulation in which labor supply is increased by 10 percent while capital remains fixed. Here we look at the version of this simulation based on the mixed version SJ.TAB of Stylized Johansen. The accurate results of the simulation (obtained by extrapolating from Euler 1,2,4-step calculations) are reported in detail in section 2.7 of GPD-1.

There are several linearized equations explicitly in SJ.TAB. Consider for example the equation in the TABLO Input file SJ.TAB for Consumer Demands:

```
Equation Consumer_demands  (All,i,SECT) p_XH(i) = p_Y – p_PC(i) ;
```

Is this equation satisfied by the simulation results (as reported in section 2.7 of GPD-1)?

Let's try this for the second commodity "s2". The relevant results are

```
p_XH("s2")         = 6.8993
p_Y                = 5.8853
p_PC("s2")         = –0.9486
p_Y - p_PC("s2")   = 6.8339
```

As you can see, these **DO NOT** satisfy the linear equation since
the LHS (Left Hand Side) = 6.8993 while the RHS (Right Hand Side) = 5.8853 – (–0.9486) = 6.8339.

This is something that often puzzles new (and some experienced) users. Although the linear equations are solved at each step (and so hold in each step) of a multi-step calculation,

usually they will not be satisfied by the accurate results.

To see why this is the case for the equation above, let's rewrite it by taking the p_PC term to the LHS, when the equation becomes

```
p_XH("s2") + p_PC("s2") = p_Y ;                                    (1)
```

In the levels, this says that Y is proportional to the product of XH("s2") and PC("s2"), that is[76]

```
XH("s2") * PC("s2") = ALPHAH("s2") * Y ;                           (2)
```

for some parameter (that is, constant) ALPHAH("s2"). We have seen from the results above that the pre-simulation levels values are

```
XH("s2")    =  4
PC("s2")    =  1
Y           =  6
```

and so ALPHAH("s2") must equal 4/6 = 0.6667.

- Suppose that the linear equation (1) above were satisfied. Then, given the simulation results p_XH("s2") = 6.8993 and p_PC("s2") = –0 0.9486, you see that

  p_Y would equal 6.8993 – 0.9486 = 5.9507.

  In that case the post-simulation values (given the pre-simulation values and the percentage changes) would be

```
XH("s2")    = 4.2670    (6,8993% larger than 4),
PC("s2")    = 0.9905    (0.9486% less than 1) and
Y           = 6.3544    (5.9507% larger than 6).
```

  These 3 values do not satisfy the levels equation (2) above since

  LHS = 4.2265 while RHS = 4.2363 (since ALPHAH("s2") = 0.6667).

  This is **one way** of seeing that the linear equation (1) should not be satisfied.[77]

- A second way of seeing that the linear equation (1) should not be satisfied is as follows. It is easy to see that the **exact** percentage change in a product A*B is

  p_A + p_B + (p_A*p_B/100)

[76] The associated levels equation is XH(i) = ALPHAH(i)*(Y/PC(i)) as you can see from Table 3.1.1a of GPD-1, or from TABLO Input file SJLV.TAB. Certainly it is easy to see how the linear equation in the text would come from this levels equation.

[77] If you substitute in the post-simulation levels values for XH("s2"), PC("s2") and Y as shown in Table 2.7a or Table 2.7b of GPD-1, and use ALPHAH("s2")=0.6667, you will see that the levels equation (2) is indeed satisfied.

if p_A and p_B are the percentage changes in A and B respectively.[78] Thus the **exact** connection between the linear variables, as derived from levels equation (2), is

```
p_XH("s2") + p_PC("s2) + [p_XH("s2")*p_PC("s2")/100] = p_Y      (3)
```

This is, of course different from the linearized equation (1) or the linearized equation Consumer_demands in the TAB file, since the term inside the square brackets [] is not included in the linearized equation in the TAB file SJ.TAB.[79] This is a **second reason** why the linear equation in the TAB file should not be satisfied by the accurate results.

The thing to take away from this discussion is that it would be a **bad thing** if the linearized equations were satisfied by the results of the simulation (see the first explanation above). It is a **good and desirable feature** that the linearized equations are not (in general) satisfied exactly by the accurate percentage change results of a simulation.[80]

Remember that the aim of the whole simulation is **to solve the non-linear levels equations**, not to solve the approximate linearized equations (as written in the TABLO Input file).[81]

---

[78] For example, suppose that originally A=1=B and suppose that p_A=5 and p_B=3. Then the new values of A and B are 1.05 and 1.03 so that the exact new value of A*B is 1.05*1.03=1.0815, which is an increase of exactly 8.15 percent. Notice that 5+3+(5*3/100) is also exactly equal to 8.15, as claimed.

[79] This highlights an important fact about the linearized equations. They are only approximations to the exact connections between the different percentage-change variables. For example, they do not include terms like the [] product term in equation (3), which is an exact relation between the percentage changes. Although we can write down an exact connection between the percentage change variables in the simple example in the text, it would be very difficult, if not impossible, to do so for other more complicated levels equations in GE models (for example, CES functions).

[80] Another aspect of this is that the linearized equations are satisfied exactly by the Johansen results (that is, the 1-step results). But these are only approximations to the accurate results.
We do not mean to indicate that no linear equations are satisfied by the accurate results. Indeed the equation Consumer_demands is satisfied for commodity 1 (s1). This is because p_PC("s1") is always zero in each step of the multi-step calculation since PC("s1") is fixed at 1 by the Numeraire equation in the model.

[81] Contrast the above with the Johansen results for this simulation. The Johansen results are easily obtained by multiplying the values in column 1 of Table 2.12.3 of GPD-1 by 10. [The first column there shows the results of a 1 percent increase in labor.]
The Johansen results satisfy the linearized equations exactly but do not satisfy the underlying levels equations accurately. For example, linear equation (1) in the text is satisfied exactly:

```
p_XH("s2") + p_PC("s2") = p_Y ;      (1)
7              +  (-1)          = 6
```

However the corresponding levels equation

```
XH("s2") * PC("s2") = ALPHAH("s2") * Y ;     (2)
```

is only approximately satisfied since
LHS = 4*(1 + 7/100) * (1 + (-1)/100) = 4.2372 while RHS = 4/6 * 6 * (1 + 6/100) = 4.24 .

6.3 Assertions

An Assertion statement requests the software to check conditions you expect to hold.

Assertions are introduced in section 3.14 of GPD-2. As explained there, assertions are tested either on the initial (that is pre-simulation) data or else on this and on the updated data produced at each step in a multi-step calculation.

If an assertion is not satisfied, the software tells you the element names (or numbers if names are not available) each time it fails.[82] For example, if the assertion

```
ASSERTION # Check no negative DVHOUS values # (all,c,COM) DVHOUS(c) >= 0 ;
```

fails for commodity "wool" you will see the message

```
%% Assertion 'Check no negative DVHOUS values' does not hold .
 (quantifier number 1 is 'wool')
```

(and once for each other such commodity 'c' where the assertion fails) and then the message

```
Assertion 'Check no negative DVHOUS values' does not hold.
```

If an assertion fails on step two or later of a multi-step calculation, you may have more difficulty in finding out the values of the coefficient(s) in question. To see the values of coefficients involved at step two or later, put the statement "**dws = yes ;**" in your Command file (see section 6.1.9). This causes DISPLAYs and WRITEs to the TERMINAL to be done at all steps (but not writes to other files).

By default, if an assertion fails, the program stops after informing you of the names of the elements where it fails. You can change this behaviour via the following statement in your Command file.

`Assertions = YES|no|warn ;` ! default is "yes"

If you include "assertions = no ;", assertions are not checked.[83]

If you include "assertions = warn ;", assertions are checked, but a failure results in a warning only. The first few warnings are shown in detail on the log file.

A summary at the end of the run tells you how many assertion failures there have been (if any). You can check if there have been any by searching for "assertion failure" in your log file.

Example from ORANIG01.TAB

In ORANIG01.TAB (see section 1.4.1 of GPD-8) are formulas to calculate DIFFIND and DIFFCOM. If the data base is balanced, these values should be zero. However, with the sort of rounding that goes on when arithmetic calculations are done on computers, and remembering that GEMPACK programs only produce results which are accurate to about 6 figures (see section 6.8), it is not reasonable to require that these values are exactly zero. Hence it would be unwise to include the Assertions

```
! Unrealistic assertions
Assertion # DIFFIND values zero # (All,i,IND) DIFFIND(i) = 0 ;
Assertion # DIFFCOM values zero # (All,c,COM) DIFFCOM(c) = 0 ;
```

As you can see from the DIFFCOM(c) values as reported in section 6.8.2, these Assertions would fail. Instead, you will find the following more modest and realistic Assertions in ORANIG01.TAB.

```
Assertion ! if below not true, program will stop with message !
 # DIFFIND = V1TOT-MAKE_C = tiny # (all,i,IND)
                      ABS(DIFFIND(i)/V1TOT(i)) <0.001;
 # DIFFCOM = SALES-MAKE_I = tiny # (all,c,COM)
                      ABS(DIFFCOM(c)/SALES(c)) <0.001;
```

It is reasonable to expect that these ratios are small.

---

[82] This was new for Release 6.0. [Release 5.2 software did not show this.]

[83] The Command file statement "NAS = yes ;" is an alternative to "assertions = no ;". This corresponds to the interactive option NAS (see section 14.4) shown when GEMSIM and TABLO-generated programs run.

6.4 Range Tests

Coefficients and levels variables may naturally have a certain range of acceptable values. For example, Coefficients representing dollar values should always be at least zero.

When Coefficients or Levels Variables are declared in the TABLO Input file, qualifiers such as (GE 0) can be used to specify acceptable ranges of values for them (see sections 3.3 and 3.4 in GPD-2). For example

```
Coefficient (GE 0) (All,c,COM) DVHOUS(c) ;
```

These range tests are carried out when GEMSIM or the TABLO-generated program runs. They affect the run of the program in the following two ways.

- When the data is read or updated, tests are carried out to check if the values are in the specified range. A run-time error will normally occur if one or more of the values is outside the acceptable range.
- If you are carrying out a simulation using user-specified accuracy (often called automatic accuracy – see section 7.4), the software uses these tests to decide if the current subinterval length is too large. [If one or more updated values are outside the specified range, the software automatically repeats the current subinterval, using a shorter subinterval length.]

Specifying acceptable ranges, and having the software carry out these range tests, ensure that you will not report simulation results based on a run where one or more of the pre-simulation or updated values is outside its acceptable range.

The typical case that motivates this is where some sector is declining as a result of the simulation, so that the level of its output is getting smaller, possibly close to zero (as in Figure 6.4). In such a case, the straight-line approximations used in Euler's method (or Gragg or midpoint) can cause the output level to become negative after one or more steps. [In Figure 6.4, the 2-step Euler result over the subinterval shown is indeed negative.]

If this happens, the software realises as soon as it updates the value to be negative that this should not happen.

- If you are using user-specified accuracy (see section 7.4), which means that the program is controlling the subinterval length, the program abandons the current subinterval and does it again, this time with a shorter subinterval length. The basic idea is that the software uses this extra information about the coefficients to make sure that subintervals are not too long.
- If you are not using user-specified accuracy (that is, you have specified the number of subintervals and the number of steps in each subinterval), the software will report the error (it will say which parts of which coefficient have gone out of range) and will then normally stop with an error. In this case you will need to rerun the simulation choosing more subintervals and/or steps.

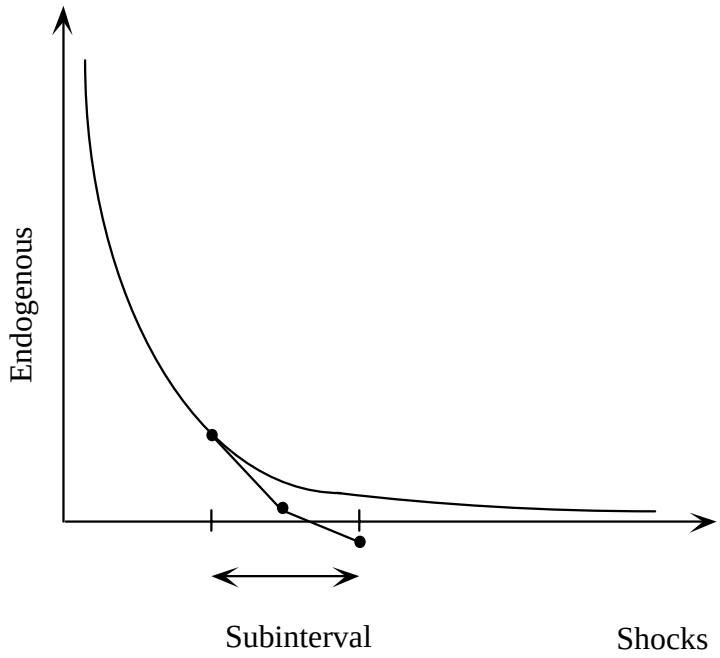


Figure 6.4 : Levels Value can become Negative if Subinterval is Too Long

6.4.1 How to Specify the Allowed Range for Values of a Coefficient or Levels Variable

This is done when you declare the Coefficient or Levels Variable in your TABLO Input file. You can use Coefficient or Variable qualifiers which specify acceptable range of values. For example,

```
Coefficient (GE 0) (all,c,COM) DVHOUS(c) ;
Variable (Levels, GE 0) (all,c,COM) DVHOUS(c) ;
```

Either of these specifies that DVHOUS(c) values should always be at least equal to zero.

When such a qualifier is used with a Levels Variable, the range restriction actually applies to the Coefficient automatically associated with this Levels Variable (see section 2.2.2 of GPD-2).

Allowed qualifiers are of the form

<operator> <real-number>

where <operator> must be one of GE,LE,GT,LT or their symbol

version

GE or >=
LE or <=
GT or >
LT or <

and <real-number> can be any real number (with or without a decimal point).

For example,

GE -1.1

LT 10

At most two such qualifiers are allowed with each declaration, namely at most one of GE,GT and at most one of LE,LT. [For example, the two qualifiers (GE 0, GT –1) are not allowed since one or other would be redundant.] An example with two such qualifiers is

```
Coefficient (GE 0, LE 1) (all,c,COM) SHARE(c) ;
```

These qualifiers given in brackets after the keyword "Coefficient" can only be attached to REAL Coefficients; they cannot be attached to INTEGER Coefficients.

We suggest that, normally, these qualifiers should be attached to Coefficients which are READ and/or UPDATEd (though the software allows them to be attached to any Coefficients).

Of course they can also be attached usefully to any Levels Variable (since these are always updated – see section 2.2.2 of GPD-2).

6.4.2 Tests Carried Out

(a) Test of Initial Values

These range qualifiers are used to test the values of the Coefficient in question at the end of the reads, formulas etc on the first step of the first subinterval of a simulation. [Note that the testing is not done immediately some or all of the values of the coefficient are read or assigned via a formula. The testing is only done once all reads and formulas are done.]

All Coefficients are tested even if one fails.[84]

If one or more fails, the program stops after all of the reads, formulas etc.

This testing is always carried out, even if user-specified accuracy is not being used.

For these tests, the range qualifiers act similarly to ASSERTIONs which are tested only on the first step of the first subinterval.

(b) Test of Updated Values

These range qualifiers are used to test the UPDATEd values of the coefficient in question (at each step and whenever its values are extrapolated).

If an updated value falls outside the specified range,

- the current subinterval is redone with a shorter length if user-specified accuracy is being used. In this case the simulation continues.
- a fatal error occurs if user-specified accuracy is not being used.

[84] Prior to Release 7.0, if TABLO Code option LMC (Low-Memory Code) was selected (see section 5.1.3 of GPD-2), the checking was not carried out for any Coefficients which are not in memory in the TABLO-generated program (this is usually real coefficients with 2 or more arguments).

6.4.3 An Example

Consider the experiment called E1 in the GTAP application about abolishing the Multi-Fibre Agreement (see chapter 10, written by Yongzheng Yang, Will Martin and Koji Yanagishima, of the GTAP book, Hertel (1997)). This is supplied in the GEMPACK examples as Command file GC10E1.CMF.

Implement GTAP61 as described near the start of section 7.1.1.

Case 1. Range Errors Produce Only Warnings

Look at the Command file GC10E1.CMF. You will see that this extrapolates from Gragg 4,6,8-step calculations over one subinterval ("subintervals = 1 ;").

Run this simulation using Command file GC10E1.CMF. Check the LOG file GC10E1.LOG produced. Notice the following lines at the end of the LOG file.

```
 (The program has completed without error.)
 (There were         2 warnings.)
 (If you have a LOG file, search for '%%W' to see them.)
 (The last was:
  'Updated value out of range')
```

The last reported warning "Updated value out of range" should suggest a problem to you.

Notice also, a few lines above, near the end of the LOG file, the lines

```
 %%WARNING. There have been 5 range-check warnings.
  [Search for '%% Test that ' in your log
    file to see the first few of each type of these warnings.]
  These warnings relate to the following types of values:
 Updated
```

Go to the top of the LOG file and search for "%% Test that" as suggested. The message at the first occurrence is

```
 %% Test that updated values of VIMS GE 0 fails
 (Value is -3.7447128)
 (quantifier number 1 of coefficient 'VIMS' is 'CLG')
 (quantifier number 2 of coefficient 'VIMS' is 'AUS')
 (quantifier number 3 of coefficient 'VIMS' is 'NAM')
```

From just above this in the LOG file, you can see that this happens during pass 3 of a 5-pass calculation. [This is during the 4-step Gragg calculation which actually does 5 passes or calculations – see section 12.2.] This warning is given since Coefficient VIMS is declared in GTAP61.TAB via the statement

```
Coefficient (ge 0)(all,i,TRAD_COMM)(all,r,REG)(all,s,REG)
    VIMS(i,r,s) # imports of i from r to s valued at domestic mkt prices #;
```

The qualifier "(ge 0)" indicates that the values of this Coefficient should never be negative.

If you continue to search in the LOG file for "%% Test that" you will see other instances of Coefficients going out of their expected ranges.[85]

Despite all of this, the simulation has completed without error. This is because (as is documented in section 6.4.4 below) the default is to give only warnings when a Coefficient goes out of range. If you think this is unsatisfactory, you can turn these warnings into fatal errors (as perhaps they should be by default) by putting the statement

```
Range test updated values = both ;
```

[85] If you used Gragg 2,4,6-steps instead of 4,6,8-steps, the LOG file would indicate 15 warnings about Coefficients going out of range, but only the first 5 would be shown explicitly in the LOG file. This is because details are suppressed in the LOG file after the first few warnings.

into the Command file. We have done this in Command file GC10E1RT.CMF (where the "RT" indicates "Range Testing") – see Case 2 below.

Case 2. Range Errors Are Fatal Errors

Look at the Command file GC10E1RT.CMF. The only difference between this Command file and GC10E1.CMF used above is that the statement "Range test updated values = both ;" has been added.

Run the simulation using Command file GC10E1RT.CMF and check the LOG file GC10E1RT.LOG. You will see that the simulation ends with an error. The following messages are near the end of the LOG file.

```
 Updated values of at least one coefficient are
  not in the required range.
 (See the earlier message(s) referring to "test that updated value".)

  (ERROR RETURN FROM ROUTINE: Main Program)
  (E-Updated value out of range)
```

If you go to the top of the LOG file and search for "test that updated value" you will see that there are 5 reports of values of Coefficients going out of range.

What can you do to fix up this problem? You could try to increase the numbers of steps until (hopefully) nothing goes out of range. That would work if you took enough steps.[86] However, a simpler way around this problem is to let the software do the hard work. You can do this by specifying automatic accuracy (see section 7.4), which we ask you to do in Case 3 below.

Case 3. Automatic Accuracy

Look at the Command file GC10E1AA.CMF (where "AA" indicates "Automatic Accuracy"). The only difference between this Command file and GC10E1.CMF used in Case 1 above is that the statement

```
Automatic accuracy = yes ;
```

has been added.

Run the simulation via Command file GC10E1AA.CMF and check the LOG file GC10E1AA.LOG. You will see that the simulation completed without error. The LOG file indicates that there was 1 range-check warning. If you go to the top of the LOG file and search for "%% Test that" you will see that the values of VIMS("CLG","AUS","NAM") went out of range as before. But this time all that happens is that the program now takes responsibility and redoes the subinterval with a shorter length. Accordingly, this simulation is now completed in 2 subintervals. [To see this, go to the top of the LOG file and search for "++>". You will see that subinterval 1 is redone with length 0.6, which completes satisfactorily without any Coefficient going out of range. Then subinterval 2 has length 0.4 of the whole simulation.][87]

Clearly using automatic accuracy is the best alternative for this simulation. Then the software can ensure that the values of all Coefficients stay in the range required by the "(ge 0)" qualifiers associated with many of the Coefficients in GTAP61.TAB.

This example is a good illustration of the value of specifying acceptable ranges of values for the Coefficients in your model. It is especially powerful when used in conjunction with automatic accuracy.[88] In general, if a simulation shows range test warnings, either increase the numbers of steps until the warnings disappear or else (a better way usually) use automatic accuracy.

You may have noticed the statement

```
Start with MMNZ = 210000 ;
```

---

[86] For this simulation, Gragg 6,8,10-steps would work.

[87] Our automatic accuracy Command file GC10E1AA.CMF is a little unusual in specifying "subintervals = 1 ;". We have done this in order to illustrate what happens when one Coefficient goes out of range. When doing automatic accuracy, the default is to try 2 subintervals (see section 7.4).

[88] The results of this application, as reported in chapter 10 of Hertel (1997), are not significantly different from the more accurate ones obtained via Command file GC10E1AA.CMF. The authors of that chapter did not have access to "GE 0" qualifiers when they carried out their application.

in the Command file GC10E1.CMF. See section 13.3 for details about this.

6.4.4 Associated Statements in Command Files

The following statements can be used to control which initial and updated values range tests are applied to.

`range test initial values = yes|no|warn ;`

[For example, if you don't want range testing of initial values, put the statement "range test initial values = no ;" in your Command file.]

`range test updated values = updated|extrapolated|both|no|warn ;`

Here

- *updated* only tests values when a coefficient has been updated (but not after extrapolation).
- *extrapolated* only tests values immediately after extrapolation (but not after updating).
- *both* tests values at both times (updated and extrapolated).
- *no* turns off both sorts of tests.
- *warn* outputs a warning for the first few instances, but the simulation carries on as if these warnings had not been given (that is, as if "no" had been selected).

The defaults and meanings of these are slightly different depending on whether or not automatic accuracy is being done. Details about these statements in the automatic accuracy case can be found in section 7.4.6.

When not doing automatic accuracy

When you are not using automatic accuracy,[89] the default value is "WARN" in each case.

When not doing automatic accuracy, if one of the values is out of range,

(i) if the relevant "*range test* …" is set at "WARN", then a warning is shown for the first few instances but the simulation carries on as if these warnings were not given.

(ii) if the relevant "*range test* …" is set at "YES", then the simulation ends with a fatal error after the value out of range is shown (in the log file). Possibly several out-of-range values of the same type (initial, updated or extrapolated) may be shown before the program stops.

(iii) if the relevant "*range test* …" is set at "NO", the relevant testing is not done.

We recommend that you set these to "YES" so that you do not report simulation results in which a value goes out of range. [If a value goes out of range, you can increase the number of steps or use automatic accuracy to keep the values in range.]

(a) If "*range test updated values = both ;*", then both updated and extrapolated values out of range are fatal errors.

(b) If "*range test updated values = updated ;*", then updated values out of range are fatal errors, and the software merely warns about extrapolated values out of range.

(c) If "*range test updated values = extrapolated ;*", then extrapolated values out of range are fatal errors, and the software merely warns about updated values out of range.

The only difference between setting one of these to "NO" or to "WARN" is that in the latter case you see individual warnings (giving the actual value and the arguments if relevant) in the log file.

If any range checks are set to be warnings only, a summary at the end tells how range check failures there have been (if there are any). You can check if there have been any by searching for "not in the required range" in your log file.

[89] Range testing when user-specified accuracy is not being carried out was introduced in Release 6.0-001.

6.5 Transfer Statements

Full details about TRANSFER statements in TABLO Input files and their operation when GEMSIM and TABLO-generated programs run can be found in sections 3.15 and 4.12 of GPD-2. You can use XTRANSFER statements in your Command file, as explained in section 4.12.1 of GPD-2.

6.6 TABLO-like Statements in Command Files

When running GEMSIM or TABLO-generated programs, you can put TABLO-like statements in Command files to

- declare SETs,
- make SUBSET declarations,
- declare extra logical FILEs,
- carry out extra WRITEs, DISPLAYs or TRANSFERs.

These extra statements are carried out **as if they were appended to the end of the original TABLO Input file.**[90] In each case the syntax and semantics are identical to that in TABLO Input files except that

- keywords are as in TABLO Input files except that an **'X' (for eXtra)** is added at the start. That is, the keywords for these statements in Command files are XSET, XFILE, XSUBSET, XWRITE, XTRANSFER and XDISPLAY.
- each statement must start with a keyword.
- there must be at least one space after the keyword before a qualifier.
  [For example, "xfile(new)" will not be recognised and should be rewritten as "xfile (new)".] See section 6.6.2 below for more details.
- the Command file syntax for comments (rest of line after single '!' is a comment – see section 2.7) is used instead of the TABLO Input file syntax for comments (in which text between two exclamation marks ! is a comment – see section 4.1.4 of GPD-2). [TABLO-like comments consisting of text between two exclamation marks '!' or between strong comment markers '![[!' and '!]]!' (see section 4.1.5 of GPD-2) are not allowed in extra statements on Command files.]

There are two main reasons why we allow these statements on Command files.

1. To allow you - **without having to rerun TABLO** - to examine the values of some COEFFICIENT which you did not write or display in the original TABLO Input file. You can now put XWRITE and XDISPLAY statements in your Command file to do this.

   For example in Stylized Johansen, to examine the values of coefficient ALPHAFAC, the following statements could be added to the Command file SJLB.CMF (see Figure 2.8.1 of GPD-1):

   ```
   xfile (new,text) sjout ;
   xwrite alphafac to file sjout ;
   file sjout = SJLBOUT.DAT ;
   ```

   See also the use of xwrite statements in the examples in section 6.8.2 below.

[90] TABLO-like statements in Command files were introduced in Release 5.2.

2. To allow you to define SETs which may be useful in specifying the closure or shocks in your simulation. These sets may not be important for the model in general (hence are not defined in the TABLO Input file), but may help you to specify a closure or shocks needed in some simulation. For example, consider the Command file statements

```
xset SPCOM (sheep, cars, services) ;
xsubset SPCOM is subset of COM ;
modify closure from file xxx ;
exogenous x1(SPCOM) ;
endogenous x2(SPCOM) ;
```

   These statements set up a closure in which certain components (those in the newly-defined set SPCOM) of variable 'x1' are made exogenous, being swapped for the corresponding components of variable 'x2'.

   See also the examples of XSET and SXUBSET statements for specifying shocks in section 5.5.1 above.

In general you can not read data using extra statements in Command files. The one exception is that you can read in set element names and numbers for new sets and subsets. The following example defines a new set "WoolGrain" and then uses it to modify an existing closure.

```
xfile extraset ;
xset WoolGrain # Wool and Grain commodities #
maximum size 4
read elements from file extraset Header "WOGR" ;
xsubset WoolGrain is subset of COM ;
file extraset = EXTRASET.DAT ;
modify closure from....
swap f5dom(WoolGrain) = x0(WoolGrain,"domestic") ;
```

6.6.1 TABLO-like Checks of Extra Statements

In the example above, the declaration of the file *extraset* (by the XFILE statement) must precede the set statement reading elements from this *extraset* file. This follows the usual rule that in TABLO, everything must be declared before it is used.

The TABLO-like statements are processed in the same order as they are given in (as if they were appended to the end of the original TABLO Input file). The TABLO-like extra statements are checked for syntax and semantic errors. Errors are reported to the terminal just as they are from the Check stage of TABLO (see section 2.1 of GPD-2). Since there is no Information file, they are reported in the LOG file. Error messages refer to the TABLO-like statements as "extra" statements. After this TABLO-style check, if there are no errors, the remaining Command file statements are processed in the usual way.

6.6.2 Qualifiers

In TABLO Input files, a space after the keyword before a qualifier is not necessary. For example, either of the following is allowed.

```
File  (New)  output   # Summary output # ;
File(New)  output   # Summary output # ;
```

But, in TABLO-like statements on Command files (see section 6.6), at least one space is required after the keyword before the qualifier. Thus, for example,

```
Xfile  (New)  output   # Summary output # ;
```

is allowed but

```
Xfile(New)  output   # Summary output # ;
```

will result in an error.

6.6.3 Other Points

The extra statements (XSET, XFILE, XSUBSET, XWRITE and XDISPLAY) only allow you to examine **coefficients already defined** in the TABLO Input file. Note that they do not allow you to declare new coefficients or variables or to add extra formulas or equations. Nor do they allow you to read extra data (except that they do allow you to read set element names and subset numbers for new sets and subsets, as in the example above).

Note that TABLO-like statements are not allowed on a Command file when you start a simulation from existing Equations and BCV files (via a "use equations file … ;" statement – see section 9.2.1). This is because, when you start from existing Equations and BCV files, the original data is not read, nor are writes nor displays carried out; hence there is no opportunity to carry out extra TABLO-like statements.[91]

[91] Prior to Release 7.0, TABLO-like statements were not allowed with a TABLO-generated program produced using any of the low memory options (LMC,ECS,SCS,UCS – see section 5.1.3 of GPD-2) from the Code stage of TABLO.

6.7 Coefficients Are Fully Initialised by Default

When you declare a Coefficient, TABLO-generated programs and GEMSIM set all values of the Coefficient to zero by default.[92]

You can change this default if you wish, by including a statement of the form

`initialise|initialize coefficients = YES|no|<value> ;`

in your Command file.[93]

- The default YES means that all values of all coefficients are initialised to zero.
- Option "no" means that Coefficient values are not initialised.
- If you wish to initialise all Coefficients to some value different from zero, you can specify the value in the statement. For example,

  ```
  initialise coefficients = 23.1 ;
  ```

  will result in all values of all Coefficients being initialised to the value 23.1.[94]

The difference between these is made clear in the following example.

Example. Consider the following TABLO Input file.

```
Set COM (c1-c3) ;
Set IND (i1-i4) ;
Coefficient (All,c,COM)(All,i,IND) DVCOMIN(c,i) ;
File FID ;
Read (All,i,IND) DVCOMIN("c1",i) from file FID Header "DVC1" ;
Read (All,i,IND) DVCOMIN("c2",i) from file FID Header "DVC2" ;
Write DVCOMIN to terminal ;
```

There is no READ or FORMULA setting the values of DVCOM("c3",i) for i in IND.

TABLO will not warn that the DVCOMIN("c3",i) values may not be fully initialised.[95] So, when the Write statement is carried out, the DVCOMIN("c3",i) values will be different depending on whether or not a statement "initialise coefficients = … ;" is included on the Command file.

- If no such statement is included (or if "initialise coefficients = yes ;" is included), these DVCOMIN("c3",i) values will be zero. [In these cases, the program behaves as if there were the statement
  ```
  Formula (All,c,COM)(All,i,IND)  DVCOMIN(c,i) = 0 ;
  ```
  after the declaration of Coefficient DVCOMIN.]
- If a statement "initialise coefficients = <value> ;" is included, these DVCOMIN("c3",i) values will be equal to <value>.

---

[92] This is true for Release 8.0 or later. In earlier Releases, Coefficients were not initialised. That is, before Release 8, the programs behaved as if the statement "initialise coefficients = no ;" was included in the Command file.

[93] You can use "s" or "z" in the first word.

[94] The values of Integer coefficients are obtained by converting <value> to the relevant integer. If <value> is 23.6, then integer coefficients are given the value 23 (the integer towards zero from 23.6). If <value> is –23.6, then integer coefficients are given the value –23 (the integer towards zero from –23.6).
If you wish to specify a negative value, do not leave any space between the – and the number part.
For example, "–23.6" is ok but "– 23.6" is not ok.

[95] This is because there are two partial reads into DVCOMIN – see section 4.10.4 of GPD-2.

- If a statement "initialise coefficients = no ;" is included, these DVCOMIN("c3",i) values will not be really determined. The relevant memory in the computer will contain random (that is, unpredictable) values.

At present TABLO gives no warning about possibly uninitialised coefficients if two or more partial initialisations are made. If no attempt is made to initialise a coefficient, TABLO will still give an error message. Full details about the way TABLO handles this issue can be found in section 4.10.4 of GPD-2.

6.8 How Accurate Are Arithmetic Calculations?

When GEMSIM and TABLO-generated programs do calculations based on data (for example, SUMs), they only produce results which are accurate to about 6 figures. The same is true of the utility and Windows programs (for example, when SEEHAR or ViewHAR report the row totals of an array of data).

We give two examples which highlights this.

6.8.1 Example 1 – GTAPVIEW

The standard GTAP TAB file referred to as **GTAPVIEW** provides a summary of any GTAP data set. We have provided GTPVEW61.TAB with the GEMPACK examples (see section 1.6.1 of GPD-8). To see the GTAPVIEW summary of the ACORS3X3 GTAP data set (these are the starting data files for the simulation in section 7.1.1), first run TABLO on GTPVEW61.TAB to produce output for GEMSIM, then run GEMSIM taking inputs from Command file GTPV3X3.CMF (which is discussed in section 4.1.1). This produces two output Header Array files called GTPV3X3.HAR (contains summary totals, shares etc) and GTPV3X3T.HAR (contains tax rates).

There is a lot of interesting information in the output file GTPV3X3.HAR. Please open this with ViewHAR (or use SEEHAR if you are working on a Unix machine). Look at the data at headers **AG01** and **AG02**. These provide a breakdown of GDP in each of the 3 regions, from the expenditure side (AG01) and the income (or sources) side (AG02). The row totals are the values of GDP for the regions. An important balance check is that GDP for any region should be the same from the two sides.

Check this for the first region SSA (Sub-Saharan Africa). Choose 6 decimal places in ViewHAR. You should see exactly the same total 316125.296875 at both headers, which is excellent balance.

What about for the second region EU? You should see that the value of GDP from the expenditure side is shown as 8208662.000000 while the value from the income side is shown as 8208663.750000. Is this apparent lack of balance serious? No, since you need to always remember that GEMPACK programs only calculate results accurately to about 6 figures. In this large number 8208662, the "2" is the seventh figure. So these two numbers agree to 6 figures, which is all you can expect. Hence the apparent discrepancy is not at all serious. Note also the difference between 6 figures (as explained above) and 6 decimal places. In the large numbers above, the sixth decimal place is the thirteenth figure.

The file GTPVEW61.TAB is an interesting example of a TAB file. For example, look at the set GDPEXPEND and at how the values in Coefficient GDPEXP are built up via several Formulas. This shows a neat way of arranging summary data into Coefficients and hence onto headers on a Header Array file. You may be able to adapt some of these techniques in your own work.

GTAPVIEW with Updated Data

You could also use GTPVEW61.TAB to check that GDP values from both sides are still equal in the updated data after any simulation with GTAP61.TAB. For example, you might like to try this for the GEX15.CMF simulation described in section 7.1.1 of GPD-3. As with SJLBCHK.CMF (see section 4.7.2 of GPD-1), you just need to change the statement in GTPV3X3.CMF (see section 4.1.1 of GPD-3) which tells which GTAPDATA file to read (and you should change the name of the Command file, perhaps to GTPVW-GEX15.CMF).

6.8.2 Example 2 – Checking Balance of ORANIG Data

ORANIG does not provide a separate TAB file for checking the balance. Instead the relevant calculations have been included in ORANIG01.TAB and are normally carried out each time you do a simulation with ORANIG. It is easy to carry out just these calculations without doing a simulation, as we show here.

First we look at the following statements in ORANIG01.TAB (see section 1.4.1 of GPD-8) which check some parts of the balance of an ORANIG data set.

```
Coefficient                    ! coefficients for checking !
 (all,i,IND) DIFFIND(i)  # COSTS-MAKE_C : should be zero #;
 (all,c,COM) DIFFCOM(c)  # SALES-MAKE_I : should be zero #;

Formula
 (all,i,IND) DIFFIND(i)  = V1TOT(i) - MAKE_C(i);
 (all,c,COM) DIFFCOM(c)  = SALES(c) - MAKE_I(c);

Write ! we file these numbers BEFORE the assertions below !
 DIFFIND to file SUMMARY header "DIND";
 DIFFCOM to file SUMMARY header "DCOM";
```

For example, DIFFCOM calculates the difference between two separate measures of the total sales of each commodity. As you can see from the TAB file, SALES(c) is obtained by adding up across intermediate use, capital creation, domestic use, exports, government use and stocks while MAKE_I(c) is obtained by adding the value of commodity c produced by each of the industries (as obtained from the MAKE matrix). If the data base is balanced, the DIFFCOM(c) values should be all zero, as should the DIFFIND(i) values for all industries i. Below we show you how to check this for the ORANIG data base contained in Header Array file OGOZD867.HAR.

You need to implement ORANIG01.TAB. Then you need to run GEMSIM from Command file OG01SUM.CMF. We spell out the details below.

6.8.2.1 ORANIG Example on a Windows PC

If you are working on a Windows PC, run WinGEM and make sure that both default directories point to the directory in which the ORANIG01 files are located.[96]

First run TABLO on ORANIG01.TAB using Stored-input file OG01GS.STI. To do this, after selecting **TABLO Implement** from the **Simulation** menu, click on the **Options** menu item on the TABLO form, and select item **Run from STI file** . Then click on the **Select** button on the TABLO form and select Stored-input file OG01GS.STI, which produces output for GEMSIM. Then click on the **Run** button. When this run finishes, click on the **Go to GEMSIM** button.

Then run GEMSIM by selecting Command file OG01SUM.CMF and then clicking on the **Run** button. When the run has finished, click on the **View Input/Output Files** button, select the output Summary file which is called SUMMARY.HAR and click **OK**. Look at the DIFFCOM results at header "DCOM" in SUMMARY.HAR. Are these values all zero?

As you have seen, the DIFFCOM(c) values are not all identically zero, though they are small. For example, the value for commodity number 4 **MiningExport** is 0.013672. As with GTAP (see section 6.8.1), this is nonzero because of the accuracy limitations (and does not mean that the data is unbalanced). To see this, note that DIFFCOM(c) is calculated as the difference between SALES(c) and MAKE_I(c). We need to see these values but, unfortunately, the values of these Coefficients are not written on the SUMMARY file. Below we show you how to use AnalyseGE to see the relevant values.

Click on menu item **AnalyseGE** under the **Simulation** menu of WinGEM. [Be careful to go to WinGEM's menu, not the menu on the TABLO or GEMSIM forms.] When AnalyseGE starts to run, click on the **Select/Change** button and open file **OG01SUM.CVL**. As usual with AnalyseGE, after a few seconds this will put you into a TABmate-like form in which you see the ORANIG01.TAB file. In this window, search for

[96] These files are supplied with the GEMPACK examples (see section 1.4.1 of GPD-8). The relevant files all have names beginning with O ("oh", not "zero"). For example, make a new directory C:\ORANIG and copy all files of the form O*.* from the GEMPACK examples directory to this directory.

DIFFCOM (use the Search..Find menu item). Go down a few lines in the TAB file until you see the Formula for DIFFCOM(c). As you saw before, this formula reads

```
(all,c,COM) DIFFCOM(c)  = SALES(c) - MAKE_I(c);
```

AnalyseGE allows you to see the values of the different Coefficients. For example, click on the word SALES on the RHS of this formula (say, click between the "A" and the "L"). Then **right click** with your mouse (that is, click with the button on the right-hand side of the mouse). A menu will appear. Select the first item **Evaluate (selection or coeff/var at cursor)** . This will put you into a ViewHAR-like window in which you can see the values of SALES(c) for all commodities c. Note that the value of SALES("MiningExport") is 19607.058594. [Select 6 decimal places in ViewHAR to see this value.] To get back to the TABmate form, click on menu item **Front** in the ViewHAR form and select item **Bring TABmate to Front** . Repeat to see the values of MAKE_I(c) for all commodities c, note that the value of MAKE_I("MiningExport") is 19607.044922, and then bring TABmate back to the front.

Let's pause to understand why DIFFCOM("MiningExport") is not exactly zero. As you have seen, this value of 0.013672 is calculated as the difference between 19607.058594 and 19607.044922. These two numbers agree to 6 figures (the sixth figure is the "0" just after the decimal place) but differ in the seventh figure (5 versus 4). Since the values of SALES("MiningExport") and MAKE_I("MiningExport") can only be relied on to be accurate to about 6 figures, you should not be disturbed by the fact that they differ at the seventh figure. Hence you should not be disturbed by the fact that DIFFCOM("MiningExport") is a sufficiently small nonzero number.[97]

Finally, we show you a way of getting the SALES and MAKE_I values at once. Go back to AnalyseGE in the TABmate form, click anywhere inside the Formula for DIFFCOM(c) [say somewhere inside "SALES" on the RHS] and right click. This time select menu item **Decompose RHS of this Formula (toggle first)** .You will again be put into the ViewHAR window. This time you see three rows. The first, labelled SALES, shows the SALES(c) values. The second row, labelled MAKE_I, shows the **negative** of the MAKE_I(c) values. The third row is the total of the two, which in this case is just the DIFFCOM(c) values. If you look at the **MiningExport** column you see the values we discussed above.

You could repeat the analysis above to look at DIFFCOM(c) values for different commodities c, and to look at the DIFFIND(i) values for different industries i. When you have finished, return to the TABmate form and select **File | Exit**. Respond **Yes** to the first prompt and **No** to the next two to exit from AnalyseGE.

Finally you should understand how it was possible to use AnlayseGE to look at the results of a data-manipulation calculation. Normally AnalyseGE is used to look at the results of a simulation. Here there was no simulation, just reads, formulas and writes. To check this, you should look at the Command file used to run GEMSIM. To do this, go back to the GEMSIM window which should be still open with the relevant Command file **OG01SUM.CMF** . Click on the **Edit** button on this form and look at the file. Notice the lines

```
! No simulation
simulation = no ;

! next allows Coefficients to be loaded into AnalyseGE.
!  (See section 8.6 of the Release 8.0 version of GPD-3.)
cvl file = yes ;
```

The line "simulation = no ;" tells GEMSIM not to do a simulation. But there are still plenty of actions to carry out, namely the reads, formulas and writes. The line "cvl file = yes ;" tells GEMSIM to write a so-called CVL (Coefficient Values) file which is what you loaded into AnalyseGE. This CVL file contains the values of all Coefficients in the ORANIG01.TAB file, calculated from the data file OGOZD867.HAR. See section 8.6 for more details about CVL files.

An alternative to creating a CVL file and using AnalyseGE would be to add the lines

```
xwrite SALES to file SUMMARY header "SALS" ;
xwrite MAKE_I to file SUMMARY header "MAKI" ;
```

These would write the values of Coefficients SALES and MAKE_I to the SUMMARY file at suitable headers. You could also write the values of Coefficients V1TOT and MAKE_C. These are used to calculate the DIFFIND values. [See section 6.6 for details about xwrite statements.]

---

[97] This is also relevant to the ASSERTION statement about DIFFCOM(c) in ORANIG01.TAB. See the ORANIG01 example in section 6.3 for a discussion.

6.8.2.2 ORANIG Example on a Unix machine

If you are working on a Unix machine, change into the directory in which the ORANIG01 files are located.[98]

First run TABLO on ORANIG01.TAB using Stored-input file OG01GS.STI via the command

tablo -sti og01gs.sti

This will produce output for GEMSIM. Then run GEMSIM taking inputs from the Command file OG01SUM.CMF via the command

gemsim -cmf og01sum.cmf

This will produce output SUMMARY file called SUMMARY.HAR. To see the values on this file, run SEEHAR interactively to produce output text file SUMMARY.SEE. Edit this file and look for the DIFFCOM(c) values which are at header "DCOM". Are these values all zero?

As you have seen, the DIFFCOM(c) values are not all identically zero, though they are small. For example, the value for commodity number 4 **MiningExport** is approximately 0.0136. As with GTAP (see above), this is nonzero because of the accuracy limitations (and does not mean that the data is unbalanced). To see this, note that DIFFCOM(c) is calculated as the difference between SALES(c) and MAKE_I(c). You need to see these values but, unfortunately, the values of these Coefficients are not written on the SUMMARY file.

An easy way to add these value to the SUMMARY file is to add suitable xwrite statements (see section 6.6) to the Command file OG01SUM.CMF. Edit this file and add the following lines.

```
xwrite SALES to file SUMMARY header "SALS" ;
xwrite MAKE_I to file SUMMARY header "MAKI" ;
```

Then run GEMSIM again. Now look at the SALES and MAKE_I values for commodity MiningExport (you will need to run SEEHAR again).

You should see that the value of SALES("MiningExport") is something like 19607.058 while the value of MAKE_I("MiningExport") is something like 19607.044. Now it is easy to understand why DIFFCOM("MiningExport") is not exactly zero. As you have seen, this value of approximately 0.0136 is calculated as the difference between 19607.058 and 19607.044. These two numbers agree to 6 figures (the sixth figure is the "0" just after the decimal place) but differ in the seventh figure (5 versus 4). Since the values of SALES("MiningExport") and MAKE_I("MiningExport") can only be relied on to be accurate to about 6 figures, you should not be disturbed by the fact that they differ at the seventh figure. Hence you should not be disturbed by the fact that DIFFCOM("MiningExport") is a sufficiently small nonzero number.[99]

Remember xwrite statements. They are easy ways of getting Coefficients written so that you can see their values. [Here you wrote them to the SUMMARY file. On other occasions you will find it convenient to write them to the terminal (that is, the LOG file) via statements like "xwrite SALES to terminal ;".]

---

[98] These files are supplied with the GEMPACK examples (see section 1.4.1 of GPD-8). The relevant files all have names beginning with o (lower-case letter "oh", not "zero"). For example, make a new directory and copy all files of the form o*.* from the GEMPACK examples directory to this directory.

[99] This is also relevant to the ASSERTION statement about DIFFCOM(c) in ORANIG01.TAB. See the ORANIG01 example in section 6.3 for a discussion.

CHAPTER 7

7. Multi-Step Solution Methods

This chapter contains details about how you can specify the solution procedure used to calculate the results of your simulation. It applies only to simulations carried out using GEMSIM or TABLO-generated programs (not SAGEM which can only produce an approximate, Johansen solution).

In particular this chapter tells you about alternative ways of ensuring that you get a sufficiently accurate solution of the underlying levels equations of your model.

The solution methods used are referred to as "multi-step" methods since they involve solving the linearized equations of the model several times, as explained in section 2.13.3 of GPD-1.

7.1 What Method and How Many Steps to Use ?

In order to calculate an accurate solution of the underlying (usually nonlinear) levels equations of your model, you usually need to carry out 3 separate multi-step calculations and then extrapolate.

When you extrapolate, you can use Euler, midpoint or Gragg as the solution method. We recommend Gragg (see section 12.2) unless you have good reasons for preferring one of the others. This is because Gragg usually produces more accurate results than Euler does (for the same number of steps).

Users of GEMPACK are sometimes unsure as to how many steps they should ask to be done.

We recommend separate multi-step calculations with 2, 4 and 6 steps respectively. Then check the Extrapolation Accuracy Summaries (see section 7.2 below). If the accuracy is not sufficient, increase the numbers of steps. For example, try 6, 8 and 10 steps. Again check the Extrapolation Accuracy Summaries.

Command file statements you will use to specify the solution method and numbers of steps are documented in section 7.1.2 below.

Usually the accuracy increases when you increase the numbers of steps. Occasionally this does not happen with Gragg or midpoint, as the example in section 12.6.5 below shows. If you observe that, we suggest that you revert to Euler's method. With Euler's method, the accuracy should always increase when you increase the numbers of steps. If you find that does not happen, there may be a problem with your model.[100]

Sometimes, especially when your shocks are large, or when the shocks produce substantial changes in the underlying economy, increasing the numbers of steps still does not produce sufficiently accurate results. There are two things you can try.

- You can increase the number of **subintervals**. The ideas behind this are described in section 7.3 below. Basically, you will usually get more accurate results if you use 6,8,10 steps over 5 subintervals than if you use 30,40,50 steps over a single subinterval (and the CPU times are approximately the same). So if you find that you are taking large numbers of steps (say, over 20 or 30), consider using several subintervals instead.
- You can let the software decide how many steps (and subintervals) to take. That is, you can use what we call **automatic accuracy** or **user-specified accuracy**. The basic idea is that, instead of telling the software how many steps, you tell it how accurate you want the results to be. [For example, you may say that you want 90 percent of the results to be accurate to at least 4 figures.] You can find full details about this in section 7.4 below.

We illustrate these general points below via a standard GTAP application.

[100] If you find that this problem persists, please contact the GEMPACK developers.

7.1.1 Example – GTAP Liberalization Simulation

Here we consider a fairly standard liberalization simulation with the GTAP model to illustrate the points above. This simulation uses version 6.1 of GTAP.TAB (called GTAP61.TAB as distributed with the GEMPACK examples – see section 1.6.1 of GPD-8). The starting data is the 3 region [Sub-Saharan Africa (SSA), European Union (EU) and Rest of World (ROW)], 3 commodity [food, manufactures (mnfcs) and services (svces)] aggregation of Version 4P of the GTAP data – these are the files GDATC3X3.HAR, GSETC3X3.HAR and GPARC3X3.HAR distributed with the GEMPACK examples. The basic Command file is GEX15.CMF distributed with the GEMPACK examples.[101] We suggest that you work through the steps below on your computer.

To prepare, you need to run TABLO.

(i) If you have a Source-code version of GEMPACK, take inputs from Stored-input file GTAP61TG.STI to produce the TABLO-generated program GTAP61.FOR, and compile and link to produce executable image GTAP61.EXE. Then carry out the simulations described below by running GTAP61.EXE.

(ii) If you have an Executable-image version of GEMPACK (or the Demonstration version), run TABLO taking inputs from the Stored-input file GTAP61GS.STI to produce GEMSIM Auxiliary files GTAP61.GSS and GTAP61.GST. Then carry out the simulations described below by running GEMSIM.

Now run the simulation.

- First carry out the simulation using Command file GEX15.CMF. This uses Gragg's method and extrapolates from separate 2,4,6 step calculations. The relevant statements in the Command file are
  ```
  method = gragg ;
  steps = 2 4 6 ;
  ```
  Look at the LOG file (GEX15.LOG) and search for "(Extrapolating – Cumulative results.)".
  You will see the Extrapolation Accuracy Summary (see section 7.2) for the variable results just below this. Go down a few lines until you see "Below are for levels values of percent-change and change results". Notice that most results are judged accurate to 6 figures (approximately 1431 of the results), and only a very small number are judged accurate to 0, 1 or 2 figures (1,1 and 16 results respectively).

- Now carry out the same simulation using Command file GEX15E.CMF. The only difference is that this uses Euler's method (rather than Gragg). Extrapolation is still based on separate 2,4,6 step calculations. The relevant statements in the Command file are
  ```
  method = euler ;
  steps = 2 4 6 ;
  ```
  Look at the LOG file (GEX15E.LOG) and find the Extrapolation Accuracy Summary for the cumulative results as above. How does the accuracy compare with that for the Gragg case above? Notice, for example, that now only about 639 results are judged accurate to 6 figures (compared to 1431) while about 161 results are now judged accurate to only 2 figures (compared to about 16 in the Gragg case). This illustrates the general point that, usually, Gragg produces more accurate results than Euler does (for the same numbers of steps).

- What about several subintervals? To see this, carry out the simulation using Command file GEX15I.CMF which uses Gragg' method again (still 2,4,6 steps), and asks for 4 subintervals via the extra statement
  ```
  subintervals = 4 ;
  ```
  Again look at the LOG file GEX15I.LOG. This time there are separate Extrapolation Accuracy Summaries after each subinterval. You need to go to near the end of the file to the section headed "ACCURACY SUMMARY FOR OVERALL RESULTS". This section summarises the accuracy of the results after all 4 subintervals (see section 7.3.1). Compare the report with that from the first case (GEX15.CMF) above. You will see that the results are slightly more accurate. For example, about 1644 results are accurate to 6 figures (compared to about 1431 with one subinterval). In this case, the extra accuracy gained from the 4

[101] This is essentially Example 15 from the GTAP Hands-on document Horridge and Pearson (2002). This simulation can be carried out using RunGTAP by selecting version ACORS3X3 and loading experiment EX15. However we encourage you to carry out the simulation directly using the Command files described in the text to make sure that you are carrying out the simulation as described in the text. (Either work at the command prompt or else work via WinGEM.)

subintervals is not overwhelming. This is because the single subinterval results from GEX15.CMF are already fairly accurate.

- What about automatic accuracy? When you use this, you specify what percentage of the results (for the variables or the updated data) should be accurate to a certain number of figures. For example, requiring that at least 90 percent of the variable results be accurate to at least 5 figures seems reasonable for this simulation.

  - ❖ To see how that works, carry out the simulation using Command file GEX15A1.CMF which uses Gragg's method (still 2,4,6 steps) and asks that at least 90 percent of the variable results be accurate to at least 5 figures. The relevant statements in the Command file are

    ```
    automatic accuracy = yes ;
    accuracy figures = 5 ;
    accuracy percent = 90 ;
    accuracy criterion = solution ;
    ```

    Again look at the LOG file GEX15A1.LOG. Again go to near the end of the file to the section headed "ACCURACY SUMMARY FOR OVERALL RESULTS". This section summarises the accuracy of the results after all the whole calculation. Compare the accuracy with the single subinterval (GEX15.CMF) and the 4 subintervals (GEX15I.CMF). You will see that the accuracy of this automatic accuracy case GEX15A1.CMF lies somewhere between these two. [For example there are 109 and 1616 results accurate to 5 and 6 figures in GEX15A1 while these numbers are 202 and 1431 for GEX15 and 123 and 1644 for GEX15I.] How many subintervals were used in the GEX15A1 case? To see this, go to the top of the LOG file GEX15A1.LOG and search for "++>" which is used to mark the start of each subinterval. You will see that just two subintervals were used.[102]

  - ❖ What happens if you ask for more accuracy – say ask for 6 figures instead of the 5 required above? To see this, carry out the simulation using Command file GEX15A2.CMF. The changed statement in the Command file is

    ```
    accuracy figures = 6 ;
    ```

    Check the overall accuracy as above, and compare it to the accuracy for the previous simulations. How many subintervals were used this time? You should be able to see that subinterval number 1 had to be redone with a shorter length (that is, with a smaller fraction of the total shocks) since the first time the accuracy was not as required. You will see that the accuracy is rather similar to the 4-subinterval case GEX15I.CMF, which is not surprising since GEX15A2.CMF also used 4 subintervals. The significant difference is that, this time the software decided how many subintervals (in order to get at least 90 percent of the results accurate to 6 figures) whereas, previously the number of subintervals was specified in the Command file GEX15I.CMF.

- Automatic accuracy must be used with care. If you increase the desired accuracy too much, the simulation can take a very long time to complete.

  - ❖ To see this, try running the same simulation via Command file GEX15A3.CMF. This requires at least 91 percent (rather than 90 percent) of the variable results to be accurate to at least 6 figures. This time, you can see from the LOG file that 16 subintervals are required.[103]

  - ❖ The liberalization carried out in the simulation GEX15. CMF above is only a partial one in the sense that the European Union is the only region liberalizing and only food (not the other commodities) is being liberalized. If you carried out a full liberalization simulation (all regions and all commodities) the software would have to work a lot harder (that is, would take more subintervals) to achieve a given level of accuracy. Indeed you might well find in that case that requiring at least 90 percent of the results to be accurate to 6 figures is not achievable.

---

[102] Two subintervals is the default (see section 7.4 below). If the accuracy had not been met on either of these subintervals, the subinterval would have been made smaller and that subinterval repeated, as explained in more detail in section 7.4 below.

[103] The reported overall accuracy with 91 percent as the criterion is not really any better than that when 90 percent was the criterion. This is partly because the criterion must be satisfied in each subinterval (rather than overall). Having more subintervals decreases the overall accuracy confidence. If you want to see how asking for too much accuracy can increase the length of the simulation, try GEX15A2.CMF replacing 91 percent by 92 percent. You will find that the simulation requires well over 100 subintervals.

You will gain experience about these issues as you work more with your model. Some models usually do not need several subintervals or automatic accuracy. For some other models (where shocks often have a large effect on the underlying economy), using automatic accuracy (with sensible accuracy requirements) is the norm. For example, using automatic accuracy is not usually necessary with the ORANI-G model whereas it often is with the GTAP model.

We give more details about these issues in the rest of this chapter. In section 7.1.2 we document the Command file statements for solution method and number(s) of steps. In section 7.2 we give more details about Extrapolation Accuracy Summaries. In section 7.3 we give details about several subintervals, while in section 7.4 we explain and document the automatic accuracy procedures.

7.1.2 Command File Statements for Method and Steps

There are four possible solution methods which can be used for multi-step simulations: Johansen, Euler, midpoint or Gragg's method. If you do not specify a method in your Command file, the default method, Gragg's method will be used.

The form of Command file statement is

method = <method_name> ; ! Default is gragg

where <method_name> can be Johansen, Euler, midpoint or Gragg.

For example method = Euler ;

Euler's method has been introduced in section 2.13.3 of GPD-1. A similar introduction to Gragg's method and the midpoint method can be found in section 12.2. Chapter 12 discusses various aspects of solving models and how to speed that up.

For models written entirely as levels equations in the TABLO Input file, (not linearised or mixed), an alternative method is Newton's method. See section 7.5 for details.

The Command file statement listing the number of steps in a multi-step simulation is

steps = <list_of_step_numbers> ;

where <list_of_step_numbers> is a list of up to 3 integers (separated by spaces),

for example steps = 4 ;

or steps = 3 5 7 ;

There is no default unless you are using automatic accuracy when the default is 2,4,6 – see section 7.4. If you specify just one step number, a single multi-step simulation is done. If you specify two or three, extrapolation is done also.

If you are using Gragg or the midpoint method, the number of steps must be all odd or all even (see section 12.2).

To obtain information about the convergence of the multi-step methods, if you are extrapolating from 2 or 3 multi-step solutions, you can ask for an Extrapolation Accuracy file using the statement:

Extrapolation Accuracy file = yes ;

The name of the Extrapolation Accuracy file is the same as the Solution file but has a suffix .XAC. [For example, if the Solution file is SJLB.SL4, then the Extrapolation Accuracy file will be called SJLB.XAC.] See section 2.13.3 of GPD-1 for an example of extrapolation. More details about Extrapolation Accuracy files can be found in section 7.2 below.

See also section 7.3 for details about subintervals:

subintervals = <number_of_subintervals> ;

7.2 Extrapolation Accuracy Summaries and Files

When you solve your model by extrapolating from 3 separate multi-step solutions (as we recommend – see section 7.1), the software provides estimates of the accuracy of the results for the variables and of the accuracy of the updated data. Summaries of both are always given. Details for each variable can be requested. We give the details below.

7.2.1 Accuracy Estimates and Summaries For Variables

When you solve a model accurately it is really the underlying levels equations you want solved and the underlying levels quantities you want accurately. With variables, the number of accurate figures in the levels value is usually more than in the associated percentage change and this difference varies depending on the size of the percentage change, as the examples below show.

- Suppose percentage-change variable p_X increases by 3.5 percent. If the pre-simulation levels value of X is 100 then the post-simulation levels value is 103.5. Here, if p_X is accurate to 2 figures, the value of X is accurate to 4 figures. If p_X is accurate to 4 figures then X is accurate to 6 figures.
- Suppose percentage-change variable p_Y increases by 0.23 percent. If the pre-simulation levels value of Y is 1 then the post-simulation levels value is 1.0023. Here, if p_Y is accurate to 2 figures, the value of Y is accurate to 5 figures. More generally, if p_Y is accurate to n figures then Y is accurate to n+3 figures.

For each component of each variable which is a percentage-change variable, the Extrapolation Accuracy file (see section 7.2.3) reports the number of figures you can be confident of for the associated levels variable (as well as for the percentage-change variable). For example, for the Stylized Johansen simulation in the standard Command file SJLB.CMF, the p_Y line in the Extrapolation Accuracy file (see section 2.13.3 of GPD-1) shows

```
p_Y  1   6.00000    5.94286    5.91412    5.88527     CX  4 L5
```

The results are the 1,2,4-step results and the extrapolation (5.88527) based on them. The comment "CX 4" is an abbreviation meaning that you can have confidence in the extrapolated result (this is the 'CX') and that the two extrapolations (the first based just on the 1,2-step results and the second based on the 2,4-step results) agree to 4 figures (or more). Note that the agreements are reported as figures, not decimal places. (For example 123.4567 and 123.4014 agree to 4 figures, but only one decimal place.) The abbreviations (such as 'CX') used on this file are explained at the top of the file. (The first "1" in the line displayed above means that this line refers to the first – in this case, the only – component of variable p_Y.)

The "L5" at the end of this line means that you can be confident of 5 figures accuracy in the level of income Y.

i. In calculating the accuracy of the levels result, we assume that the pre-simulation levels value is 50 (rather than 100 which may seem more natural). This gives better symmetry between positive and negative percentage changes. [Suppose we started from 100. If p_X increases by 3.5 per cent, the new levels value of X is 103.5 and 2 figures in p_X translates to 4 figures in X. But if p_X decreases by 3.5 per cent, the new levels value of X is 96.5 and 2 figures accuracy in p_X only then translates to 3 figures accuracy in X.]

ii. For CHANGE variables, the linear result is already a levels result (that is, it reports the change in the levels value). Hence no "Lx" is added for components of change variables.

The Extrapolation Accuracy Summaries (output to the terminal or at the end of the Extrapolation Accuracy file) give a summary for these levels results (as well as for the linearised variables). For example, at the end of SJLB.XAC (with Euler 1,2,4-step calculations) you will find something like

```
  SUMMARY OF CONVERGENCE RESULTS                    Number   Min Figs
  ------------------------------                    ------   --------
  XMA  Three extrapolations equal to machine accuracy  3         6
  FC0  Fair confidence that the result is zero         2
  CX   Confidence in the extrapolated result          22         2

  2 results are judged accurate to 2 figures.
  4 results are judged accurate to 3 figures.
  16 results are judged accurate to 4 figures.
  3 results are judged accurate to 6 figures.

 Above is for linearised variables.
 Below are for levels values of percent-change and change results.

  1 results are judged accurate to 4 figures.
  21 results are judged accurate to 5 figures.
  5 results are judged accurate to 6 figures.

(The summary above covers the XAC-retained variables.)
```

The list of possible codes (XMA, CX etc) is shown in section 7.2.3.

The number of figures of accuracy for the levels variables can be used as the basis for automatic accuracy – see section 7.4.

If you have several subintervals (see section 7.3 below) or if you use automatic accuracy (see section 7.4 below), there are separate accuracy summaries for each subinterval and an Overall Accuracy Summary (see section 7.3.1) at the end.

Note that, if you are extrapolating from only 2 multi-step simulations, no codes (for example, "CX") appear on the Extrapolation Accuracy file and no Extrapolation Accuracy Summary is given. This is because it is not possible to estimate the size of the errors on the basis of only 2 solutions.

7.2.2 Accuracy Estimates and Summaries For Updated Data

When extrapolating, it is possible to give estimates of accuracy for each value on each updated data file in the same way as for each component of each endogenous variable. Such summaries are given (to the terminal and/or to a LOG file) for each separate data file. In addition, a summary across all updated data files is given (if there is more than one relevant file).

You can look at these summaries (as well as that for the variables) if you are trying to decide whether your simulation results are sufficiently accurate.

For example, near the end of the LOG file SJLB.LOG from the SJLB.CMF simulation in chapter 2 of GPD-1, you can see something like

```
 NEXT SUMMARY IS FOR UPDATED DATA FROM LOGICAL FILE iodata

    SUMMARY OF CONVERGENCE RESULTS                  Number   Min Figs Agree
    ------------------------------                  ------   --------------
  XMA  Three extrapolations equal to machine accuracy    3         6
  CX   Confidence in the extrapolated result             7         5

  7 results are judged accurate to 5 figures.
  3 results are judged accurate to 6 figures.
```

This estimates that, of the ten values on the updated data SJLB.UPD (see section 2.9 of GPD-1), 7 are accurate to 5 figures and 3 to 6 figures.

If there are 2 or more data files updated in a simulation, a summary for each file is given and then an overall summary for all the updated data is given.

The software uses these Extrapolation Accuracy Summaries for the data as the basis for its automatic accuracy option – see section 7.4.

If you have several subintervals (see section 7.3 below) or if you use automatic accuracy (see section 7.4 below), there are separate accuracy summaries for each subinterval.[104]

7.2.3 Extrapolation Accuracy Files

If you are extrapolating from 2 or 3 multi-step solutions, you can ask for an Extrapolation Accuracy file by including the statement

Extrapolation Accuracy file = yes ;

in your Command file. The name of the Extrapolation Accuracy file is the same as the Solution file but has a suffix .XAC. [For example, if the Solution file is SJLB.SL4, then the Extrapolation Accuracy file will be called SJLB.XAC.]

If you extrapolate from 3 multi-step solutions (as we recommend), this Extrapolation Accuracy file contains details of the separate solutions, of the extrapolated solution and accuracy estimates **for each endogenous component of each endogenous variable**.

For example, for the Stylized Johansen simulation in the standard Command file SJLB.CMF, the p_Y line in the Extrapolation Accuracy file (see section 2.13.3 of GPD-1) shows

```
p_Y  1   6.00000    5.94286    5.91412    5.88527     CX  4 L5
```

The meanings of the different parts of this line are explained in section 7.2.1 above.

The component numbers shown in each line of an Extrapolation Accuracy file are calculated as in section 5.3.

For example, the p_XC results in SJLB.XAC show

```
4    7.00000   6.95000   6.92473   6.89929   CX  3 L5
```

These are the results for p_XC("s2","s2") since that is component number 4 of variable p_XC(SECT,SECT).

Extrapolation Accuracy files can be very large since there is one line for each endogenous component. So you should be cautious about requesting one.

If you have several subintervals (see section 7.3 below) or if you use automatic accuracy (see section 7.4 below), the information about each variable is repeated for each subinterval. This can make the Extrapolation Accuracy file even larger.

Note that, if you are extrapolating from only 2 multi-step simulations, no codes (for example, "CX") appear on the Extrapolation Accuracy file and no Extrapolation Accuracy Summary is given. This is because it is not possible to estimate the size of the errors on the basis of only 2 solutions.

At the top of the Extrapolation Accuracy file is a list of all possible codes (for example "CX") and their meanings. These are:

[104] However, there is no overall accuracy summary for the updated data (as there is for the variables – see section 7.3.1).

```
 CODE      MEANING
 ----------------
  XMA      Three extrapolations equal to machine accuracy
  XRA      Three extrapolations equal to required accuracy
  C0       Confidence that the result is zero
  FC0      Fair confidence that the result is zero
  CX       Confidence in the extrapolated result
  FCX      Fair confidence in the extrapolated result
  MVC      Monotonic, results very close together
  MN0      Monotonic, results near zero
  MC?      Monotonic, appears to be converging
  MD?      Monotonic, but may be diverging
  MD!      Monotonic, but appears to be diverging
  OVC      Oscillating, but results very close together
  ON0      Oscillating, but results near zero
  OC?      Oscillating, but appears to be converging
  OD?      Oscillating and may be diverging
  OD!      Oscillating and appears to be diverging
```

The ones near the top (down to FCX) indicate satisfactory convergence, while MVC and MN0 are ok. Any code with a question mark ? indicates not satisfactory convergence, while codes with an exclamation mark ! are the worst. Note also that "oscillating" is usually worse than "monotonic".

7.2.4 Command File Statements Affecting Extrapolation Accuracy Summaries and Files

If you are extrapolating from three multi-step simulations, an Extrapolation Accuracy Summary is always output to the terminal. This is the same as the summary produced at the end of the Extrapolation Accuracy file if you request such a file. It gives information to help you decide if the solutions have been calculated sufficiently accurately for your purposes. As explained in section 7.2.3 above, the Extrapolation Accuracy file contains a brief annotation (for example, "EMA 6") about each solution. The Extrapolation Accuracy Summary summarises these.

7.2.4.1 Required Figures of Accuracy (Option RQF)

You can put a statement of the form

RQF = <integer> ;

in your Command file to set the number of figures agreement you require.[105] This is used in determining the two codes

ERA Last two results equal to required accuracy

XRA Two extrapolations equal to required accuracy

in the Extrapolation Accuracy file and on the Extrapolation Accuracy Summary. If, for example, you select 3 figures then comment "ERA" will be given to any results whose last two multi-step solutions agree to three (or more) figures (unless they are equal to machine accuracy in which case the stronger comment "EMA" will apply).

Codes "ERA" and "XRA" never appear unless you have such a statement in your Command file (or select option **RQF** – see section 14.2).

[105] Alternatively you can use option RQF – see section 14.2.

7.2.4.2 Selecting Variables on the Extrapolation Accuracy File

By default, all endogenous variables plus any you elected to backsolve for appear on the Extrapolation Accuracy file (if you request such a file) and are summarised in the Extrapolation Accuracy Summary (for variables) sent to the terminal and the LOG file. If you want to change the variables shown on the Extrapolation Accuracy file, you can do so by putting one or more statements of the form

XAC-retained <list> ;

in your Command file.[106]

This enables you to select the variables to appear on the Extrapolation Accuracy file (these variables are referred to as the **XAC-retained variables**). You can select any exogenous or endogenous variables or any you elected to backsolve for to be on the Extrapolation Accuracy file. (The Extrapolation Accuracy Summary for variables then summarises just the XAC-retained variables.)

[106] Alternatively you can use option SVX – see section 14.2. If you are running interactively (or via a Stored-input file) and you select option **SVX** at the start, you choose the XAC-retained variables after the verbal description of the simulation (which makes it the last piece of user input).

7.3 Splitting a Simulation into Several Subintervals

Normally, if you want to obtain accurate solutions to the nonlinear equations of your model, you calculate 3 multi-step solutions and extrapolate on the basis of these. Each of these moves away from the exact solution curve as it goes along (though the one taking the most steps stays closer), but the final extrapolated result is hopefully almost on the curve again. This is illustrated for 1,2,4-step solutions in Figure 7.3a below, in which X on the horizontal axis is exogenous (moving from X_0 to X_1) and Y on the vertical axis is endogenous.

In this case, you can't tell how accurate your final solution is until you see the Extrapolation Accuracy Summary at the end. In a highly nonlinear case you may calculate 40,60,80-step solutions, perhaps taking several hours, only to find that the accuracy at the end is not what you require. Then you must start all over again, choosing more steps.

This is one of the situations in which we have found it useful to be able to split the simulation into several so-called **subintervals**. In each subinterval the exogenous variables are changed by the same (levels) amount, so that, after all is done, the total change is as required. The basic idea is to go across each subinterval with 3 multi-step calculations (usually with a small number of steps) and then to **extrapolate before starting the next subinterval**. The data base is extrapolated as well as the solution. Hopefully the solution comes back very close to the exact one at the end of each subinterval. This is illustrated in Figure 7.3b for two subintervals and 1,2,4-step calculations across each one.

For example, instead of calculating 40,60,80-step solutions, you could break the simulation into 10 subintervals and take 4,6,8-step calculations across each one. The total calculation time is approximately the same.

- The main advantage is that the results from the several subinterval case are likely to be more accurate. [See section 7.1.1 above for an example.] Intuitively, the reason why several subintervals are likely to produce better accuracy for a given total number of steps is because then (hopefully) the simulation returns to close to the intended path after each subinterval (whereas a 40,60,80-step calculation for a highly nonlinear simulation can have each one far from this path towards the end). [Figures 7.3a and 7.3b above make this moderately clear since, in Figure 7.3a, even if there are 40 or so steps, the points used to start the later steps may be well away from the curve in a highly nonlinear case.] Another way of putting this is that the errors in each simulation after one subinterval are likely to be significantly less than those at the end of the whole simulation, so the extrapolated result is likely to be much more accurate.[107]
- Another (less important) advantage is that you get an Extrapolation Accuracy Summary (see section 7.2) for each subinterval. Thus, at the end of the first subinterval, you can see if the solution accuracy is what you require; if it is not, you can stop the simulation then and restart it with more subintervals or more steps across each subinterval (or both). Thus, having several subintervals may give you early warning of unsatisfactory accuracy.

To break a simulation into several subintervals, you can use the Command file statement[108]

subintervals = <number> ;

The number of steps you specify will be carried out **in each subinterval**. You should specify the shocks you intend for the whole interval; GEMSIM or the TABLO-generated program automatically breaks this up into the appropriate (smaller) shocks for each subinterval and then for each step in each subinterval.

---

[107] In a GTAP simulation with large shocks reported to us, Gragg 50,100,150 steps produced few results accurate to more than one or two figures but Gragg 4,8,16 steps over 10 subintervals produced more than adequate accuracy (and took less CPU time).

[108] Alternatively select the option

SSI Several subintervals; extrap after each

after which you will be asked for the number of subintervals.

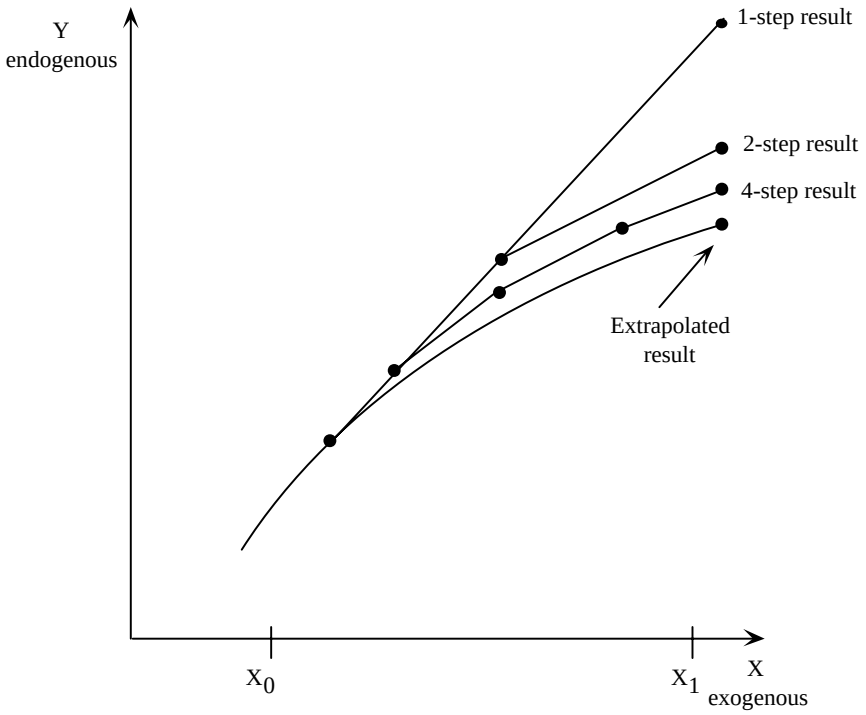


Figure 7.3a: One Subinterval

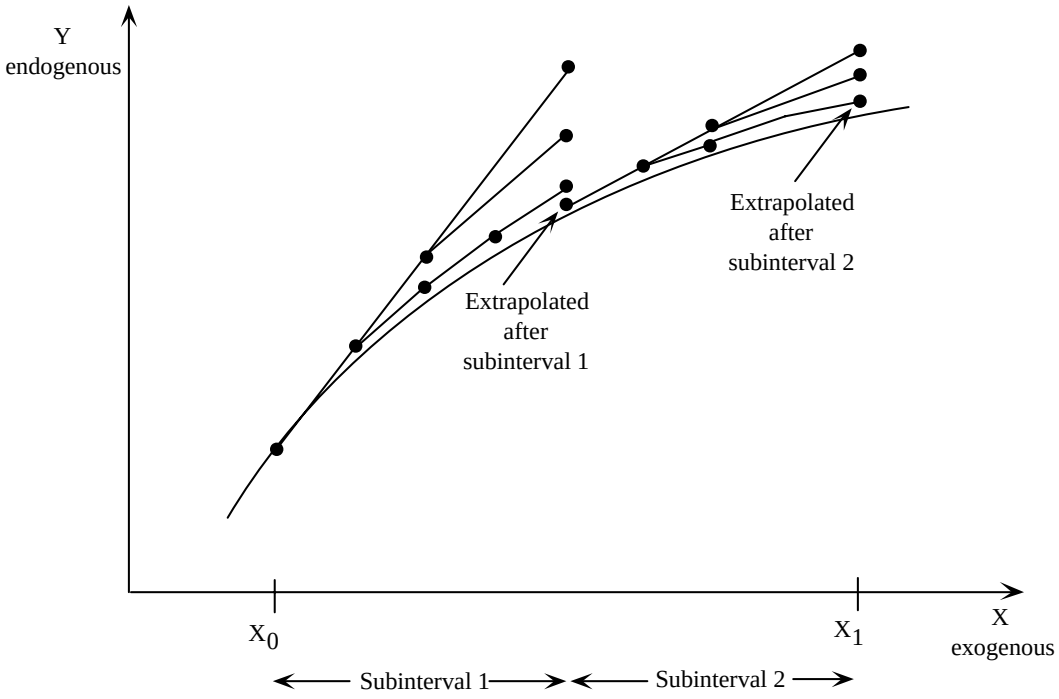


Figure 7.3b: Two Subintervals

An Extrapolation Accuracy Summary (see section 7.2) is output to the terminal after each subinterval. If you are also creating an Extrapolation Accuracy file (see section 7.2.3), the results for the variables are output to it for each subinterval; these results are the endogenous movements just across the current subinterval.

> If, for example, you have 2 subintervals and a percentage-change variable X increases by 5 per cent across the first subinterval and by 4 per cent across the second one, this means it will increase by the combination of these, namely 9.2 per cent, across the whole simulation. The Solution file will show 9.2 while the Extrapolation Accuracy file will show 5 and 4 for the two subintervals.

When several subintervals are used (with or without user-specified accuracy), FORMULA(INITIAL)s are only carried out during the first step of the first subinterval. [Values in subsequent steps and subintervals are determined from these initial values taking into account UPDATE statements.]

7.3.1 Overall Accuracy Summary When More Than One Subinterval

If you use more than one subinterval, the LOG file contains an Extrapolation Accuracy Summary for the variables in each of the subintervals. However it is not easy to combine these to obtain an accuracy summary for the whole result.

An Overall Accuracy Summary is produced when the simulation uses two or more subintervals.[109] This includes most simulations solved using automatic accuracy (section 7.4).

The Overall Accuracy Summary looks very similar to the Extrapolation Accuracy Summaries for each subinterval.

For example, suppose that there are just three subintervals. Consider a scalar variable with the following accuracy:

- accurate to 5 figures in the first subinterval with code FCX (Fair Confidence in eXtrapolated result)
- accurate to 3 figures in the second subinterval with code MC? (Monotonic, appears to be converging)
- accurate to 4 figures in the third subinterval with code OC? (Oscillating, appears to be converging)

Then the Overall Accuracy Summary will count this as being accurate to 3 figures (the lowest of the three) with code OC? (the worst of the three).

Note that taking the worst of the number of accuracy figures is a reasonable estimate **provided that the number of subintervals is not too large**. For example, suppose that, in every subinterval, the result for a scalar variable is accurate to 5 figures. If there are only 3-4 subintervals, you can be reasonably confident that the overall result for this scalar variable is still accurate to 5 figures. But if there are 100 subintervals, it is highly likely that the overall result would only be accurate to 3 or 4 figures.

7.4 Automatic Accuracy for Simulations

Automatic accuracy (sometimes called **user-specified accuracy)** has two motivations.[110]

- Firstly it allows you to specify the **required accuracy** needed in a simulation. You can specify this accuracy in advance and let the programs (GEMSIM or TABLO-generated programs) decide how many steps and how many subintervals to take to achieve the accuracy you have specified. See section 7.1.1 above for an example.
- A second advantage is that, when used in conjunction with range restrictions in your TABLO Input file (see section 6.4), automatic accuracy ensures that the **Coefficients never stray outside the specified range**. See section 6.4.3 for an example of this.

This option is only available if you use a Command file. The simulation must be a multi-step one and it must be one in which you extrapolate from 3 separate multi-step calculations.

The statements in a Command file which let you specify the accuracy are shown below.

---

[109] The Overall Accuracy Summary was introduced with Release 8.

[110] This feature was new for Release 5.2 of GEMPACK.

Automatic accuracy = yes|NO ; ! NO is the default
! use "yes" to say the you want user-specified accuracy

accuracy figures = <d> ; ! 4 is the default
! e.g. accuracy figures = 5 ;

accuracy percent = <pp> ; ! 80 percent is the default
! e.g. accuracy figures = 90 ;

accuracy criterion = DATA|solution|both ; ! data is the default
! e.g. accuracy criterion = both ;

If you specify "automatic accuracy = yes ;" then the default is to require at least 80 percent of the extrapolated data results to be accurate to at least 4 figures. [See section 7.2.2 above for Extrapolation Accuracy Summaries for the (updated) data.]

When you ask for automatic accuracy, the software solves the simulation using several subintervals, and varies the length of the subintervals to achieve the required accuracy. You can give it a hint as to how many subintervals to try by including a statement in your Command file of the form

subintervals = <number> ; ! Default is 2 (for automatic accuracy)

For example, if you suggest 10 subintervals, the software will give one-tenth of the shocks as the first subinterval and then see if the required accuracy has been achieved over this subinterval. If so, it will go on to the next subinterval (and it may increase the length of the second subinterval). If not, it reduces the length of the subinterval and re-does this first subinterval. If you don't indicate how many subintervals in your Command file, **the default is** to begin with **2 subintervals.**

You do not need to specify the **number of steps** to take across each subinterval. **By default**, the software carries out **2,4,6-step** calculations (followed by extrapolation) in each subinterval. However, if you specify different numbers of steps (for example "steps = 1 2 4 ;" with Euler's method), this many steps will be used in each subinterval. As usual, **Gragg's method** is used **by default** unless you specify another method.

Although the software varies the length of the different subintervals, it always takes the same number of steps across each subinterval when giving user-specified accuracy.

Note that the accuracy you require is achieved for each subinterval, which is not exactly the same as requiring this accuracy across the whole simulation. The Overall Accuracy Summary (see section 7.3.1) gives the best indication of the overall accuracy of your results.

The **accuracy criterion** can be either **data**, **solution** or **both**. When it is data, the Extrapolation Accuracy Summary for all data files (see section 7.2.2 above) is used to decide if the required accuracy has been achieved. If it is the solution, the Extrapolation Accuracy Summary for the levels variables (see section 7.2.1 above) is used. If you specify "both", the total of these two summaries is used.

The default accuracy criterion is the data since this is somehow the intrinsic numerical information underpinning the model. If you use the solution as the accuracy criterion, the results can be affected by how many summary variables you attach or which variables' results are taken into account (see section 7.4.1 below).

We recommend that you experiment slowly with this procedure and that you do not specify very high accuracy until you have first tried lesser accuracy (for example, tried using the default accuracy of at least 80 percent accurate to at least 4 figures). The example in section 7.1.1 is relevant in this context.

If you are using automatic accuracy and the Left Hand Side Matrix is singular in one step of a subinterval, the subinterval is redone with a shorter length. This may work around a singularity along the path from the pre-simulation values of the exogenous values to the post-simulation values.[111]

7.4.1 Specifying Which Solutions to Use

When the accuracy criterion is "solution" or "both", the solutions used in the accuracy test are the XAC-retained variables. This means that, if you wish to only take into account certain variables in the accuracy test at the end

[111] The first step of the first subinterval is not redone. [There is no possibility of working around a singularity at the start of the simulation.] This feature was introduced in Release 8.0.

of each subinterval, you can achieve this by selecting just those variables as the XAC-retained ones (see section 7.2.4.2). For example, you may be happy with the results of a certain application provided all macro results and all export volume results are sufficiently accurate; in this case, select these to be the XAC-retained variables via a statement like .

```
xac-retained   %macro   p_exports ;
```

in your Command file (assuming that p_exports is the variable denoting percentage changes in export volumes).

Note that, even if you are not saving an Extrapolation Accuracy file (that is, you do not have the statement "extrapolation accuracy file = yes ;" in your Command file), you can still select a set of XAC-retained variables to be used as the accuracy criterion.

7.4.2 Incompatibilities

Note that "automatic accuracy = yes ;" is

- incompatible with option NUD (no final updates – see sections 6.1.7 and 14.1),
- incompatible with Newton's method (see section 7.5 below),
- incompatible with the SUP and SSL statements described in section 7.7, and
- cannot be used in TABLO-generated programs which were written using any of the (now unavailable) low-memory options LMC,ECS,UCS,SCS from the Code stage of TABLO (see section 5.1.3 of GPD-2).

7.4.3 Automatic Accuracy with Models which Read Data from the Terminal

In Release 5.2, to use automatic accuracy with such models, the simulation was required to start from existing Equations and BCV files. This restriction was removed with Release 6.0.

If you have a data file none of the items on which require updating (for example, a parameters file), you cannot start from existing Equations and BCV files (see section 9.2) if you are doing automatic accuracy or have two or more subintervals.

7.4.4 Adaptive Stepsize Method

The method used to implement user-specified accuracy is simply a variant of the well-known "adaptive stepsize" method often used when solving differential equations [for example, Chapter 15 of Press et al (1986)].

7.4.5 Stopping if Subintervals Become Too Small

It may happen that your simulation cannot be solved with the specified accuracy. [This may be because you have made an error with the model or in specifying the shocks. Or it may happen because you have specified greater accuracy than is appropriate.] Then the software will continue to reduce the subinterval length. You need some way of stopping the simulation so that it does not go on for ever.

For this reason, you can specify the minimum subinterval length allowed and you can say what happens if that minimum is reached. [Normally you would want the simulation to stop with an error if the minimum is reached.] The following Command file statements can be used.

`Minimum subinterval length = * ;` ! default is 0.000001 (10^{-6})

for example,
minimum subinterval length = 0.000000001 ;
minimum subinterval length = 1.0e-9 ;
[The minimum length can be specified in decimal (eg, 0.00001) or exponential (eg, 1.0E-9) notation.]

`minimum subinterval fails = STOP|continue ;` default is "stop"

If the current subinterval has the minimum allowed length and the accuracy criterion or range test still fails, this tells whether the simulation stops at this point or continues. If it continues, the subinterval length stays fixed at the minimum allowed subinterval length even though one or more range tests are

failing. [Accordingly, continuing is not advised because your simulation results will be based on values out of the acceptable range.]

7.4.6 Controlling Which Values are Tested

Ensuring that values of coefficients do not go outside the allowed range (as specified in the TABLO Input file – see section 6.4) is an important feature of automatic accuracy.

You can control which values (initial, updated, extrapolated) are tested and what happens if a value is out of range by using the Command file statements

```
range test updated values = updated|extrapolated|both|no|warn ;
range test initial values = yes|no|warn ;
```

When you are doing user-specified accuracy, the default values are BOTH and YES respectively.[112] To get the full benefits of user-specified accuracy, you must leave these at these default values.

For example, if the relevant "range test …" is set at "WARN", a warning is shown for the first few instances out of range but the simulation carries on as if these warnings were not given. [In particular, the subinterval is not redone. So results may be based on values which are out of range.]

These range test statements can also be used when you are not doing user-specified accuracy – see section 6.4.4.

7.4.7 Checking Progress During a Run

If you are solving using user-specified accuracy, there may be several subintervals, and the run may take a long time. It can be difficult to tell from the screen output how far the simulation has progressed.

For this reason the program writes an additional file, the **Automatic Accuracy Log file**[113], which you can look at from time to time to check on the progress. This file has suffix **.aal** and the first part of its name is the same as that of the Solution file. [For example, if the Solution file is sjlb.sl4 then sjlb.aal is the name of the Automatic Accuracy Log file.] You can **copy** this file to some other file at any stage during the run and then examine its contents. However, **do not edit this file** since then, the program may not be able to update it at the end of the next subinterval which could cause the simulation to crash.

The Automatic Accuracy Log file has one line for each successfully completed subinterval. This shows how much of the simulation has been completed and when that subinterval was finished. A new line is added to the file at the end of each successful subinterval. For example, the file may contain the following lines:

```
Subinterval 1. Completed 25.000000 percent. 06:51:44
Subinterval 2. Completed 55.000000 percent. 06:52:30
Subinterval 3. Completed 91.000000 percent. 06:53:11
Subinterval 4. Completed 100.00000 percent. 06:54:02
```

The program also writes progress information to the screen (and log file) each time it begins the LU decomposition. This information looks like:

```
[Completed 13.3434 percent of whole sim. Subint 12 will do 10.5432.]
```

This tells you how much of the whole simulation has been completed before the start of the current subinterval. It also tells you how much more of the whole simulation will be completed if this subinterval is successful.

---

[112] The defaults are "WARN" if user-specified accuracy is not being used – see section 6.4.4.

[113] This file was introduced in Release 8.0.

7.5 Newton's Method for Levels Models

Newton's method is sometimes an efficient way to solve levels equations. When a reasonable starting guess can be made for the solution of the levels equations, the method converges rapidly. However if the starting solution is not a good guess, there is no guarantee that the solution will converge at all.

Newton's method can be used in GEMPACK with levels models in certain circumstances.[114] TABLO is used to add the appropriate Newton correction terms to linearised form of the levels equations. With this implementation of the model, GEMSIM or the TABLO-generated program can be used to run a simulation using the following ways of using the Newton terms:

- The shock to the Newton variable can be added to the usual shocks in an Euler solution, and/or
- the Newton shock can be applied alone in a series of Newton correction steps after a single or multistep Euler simulation.

The Newton variable is called **$del_newton** [TABLO introduces this name automatically]. For a levels equation

```
F = 0
```

the linearised equation, in terms of the change in F (written below as dF) and the Newton correction, is

```
dF = - F * $del_newton.
```

For percentage change differentiation of the levels equation

```
F = G
```

the linearised equation, in terms of the percentage changes in F and G (written as pF and pG below) is

```
pF = pG – (1 – G/F) * 100 * $del_newton
```

1. To use TABLO to add this $del_newton term to all levels equations, in TABLO you choose option
   NWT – Add Newton-correction terms to levels equations
   from the main TABLO menu (the one presented before the CHECK stage – see section 2.1 of GPD-2).
2. When you run GEMSIM or the TABLO-generated program (after compiling and linking as usual – see section 1.2 of GPD-2), you must use Euler as the solution method
3. Set the variable $del_newton exogenous. However there is no need to give the shocks for $del_newton.
4. To use the Newton method of solution when running GEMSIM or a TABLO-generated program, put **NWT = yes ;** in your Command file,[115] then continue with the simulation as usual.
5. You will be prompted for details as to whether the Newton correction is applied during and/or after each step in the multi-step simulation. You can choose to apply just the usual shocks with no Newton correction or the usual shocks and simultaneously a Newton correction. After each step you can apply none, one or several Newton corrections to improve the accuracy of the solution.

Note that, in order to user Newton's method, you must run the program interactively (or from a Stored-input file) so that you can respond to prompts after each step. Of course, you can still supply the main specifications of your simulation on a Command file.

As an example of Newton's method for the Levels model MOLV.TAB, run TABLO using the Stored-Input file MOLV-NWT.STI. The relevant files are supplied with the GEMPACK examples – see section 1.2 of GPD-8. At the Unix/Command prompt, enter

---

[114] Newton's method was introduced in Release 5.2.

[115] Alternatively, select option NWT from the initial options screen.

```
tablo -sti molv-nwt.sti
ltg  molv-nwt
```

The Executable-image MOLV-NWT.EXE produced can be run using the Stored-input file MOLVNWT1.STI.

```
molv-nwt  -sti  molvnwt1.sti
```

This runs a simulation from the Command file MOLV-NWT.CMF using Newton's method.

7.5.1 Newton's Method and ORANIF

In testing this method using a levels version of ORANIF, difficulties were encountered with zeros in the database and zero divides. Since the Newton-correction term is evaluated as part of an equation, you can not use the ZERODIVIDE defaults available for formulas. In general it is possible to use IF syntax (see section 4.4.6 of GPD-2) instead.

In the expressions for CES, CET and Cobb-Douglas functions, there are various expressions of the form A raised to the power B or involving the logarithm of A. In these expressions A must be positive, not zero. In practice this can be set up initially by setting levels variables which occur in CES, CET and Cobb-Douglas functions to at least "tiny", a small positive number. This was done even in cases where strictly speaking the levels variable would be identically zero eg if commodity c is not used at all in industry i. First the levels variable eg V1BAS was read in from a file, then a FORMULA(INITIAL) was used to change any values less than tiny.

```
FORMULA(initial) (all,c,COM)(all,s,SRC)(all,i,IND)
  V1BAS(c,s,i) = V1BAS(c,s,i) +
                 IF(V1BAS(c,s,i) lt tiny, tiny – V1BAS(c,s,i));
```

Condensation

Condensation becomes more complicated due to the addition of the $del_newton terms to each equation. To overcome this the equations for CES functions were not used in condensing ORANIF and a less condensed system of equations was solved. The increase in time caused by the increased size of the Equations matrix was offset by the fact that Newton converges faster so fewer steps could be used.

Condensation also caused division-by-zero and invalid-exponentiation problems (see sections 15.4 and 15.5) since the symbolic differentiation is not clever enough to cancel common factors in the numerator and denominator.

7.5.2 Convergence

When Newton converges, it does so rapidly – you can check convergence by checking the total of the Newton correction terms (that is, sum of the absolute values of the coefficients of $del_Newton) which is output to the terminal and/or LOG file after each Newton step. However if the starting solution is not reasonably close to the correct solution, there is no guarantee that the solution will converge at all. In fact it can go wildly wrong and end up in a region where the CES functions are not defined (eg negative prices), causing the program to crash with a Fortran error such as undefined exponentiation.

7.5.3 General advice

When specifying the method of solution in the Command file or during the simulation you must specify Euler (for example, **method = Euler ;**). Only Euler's method can be used, not Gragg or midpoint. Extrapolation does not work with Newton's method. In practice, a one-step or two-step Euler followed by 3 or 4 Newton steps gave good accuracy for various models. Solving ORANIF using a 1 step Euler followed by a 3 step Newton gave similar accuracy to 8,10,12 Gragg and extrapolation without Newton. The CPU time for the Euler-Newton was 44 secs which compares favourably with 48 secs for the fully linear version of ORANIF with 8,10,12 Gragg.

Newton's method can also be used to check the accuracy of the initial solution since the Newton terms at the first step are a calculation of the error in the equations for the original data. Information about the size of the

Newton correction terms is given even if you are not doing a simulation.[116] (Put "simulation = no ;" and "nwt = yes ;" in your Command file to see this information.)

Adding homotopy terms to levels equations (see section 7.6) may be preferable to using Newton's method in some cases. There may be convergence using the adding homotopy terms method in cases where there would not be convergence using Newton's method starting from the same initial guesses.

Levels models are available for Stylized Johansen, Miniature ORANI and ORANIF (see chapter 1 of GPD-8).

7.5.4 Newton's Method Is Not Reliable with Mixed Models

Note that ***we do not recommend Newton's method with any models except fully levels ones.***

We do not recommend the use of Newton's method with mixed models (those whose TABLO Input files contain some levels and some linearised equations). In such a case, TABLO is only able to add the Newton correction terms to the levels equations. We have found that the solution appears to converge but that it can converge to the "wrong" solution in these cases (that is, to a solution which is not a solution of the underlying levels model).

[116] This was new for Release 7.0. We are grateful to Ronald Wendner for suggesting this.

7.6 Homotopy Methods and Levels Equations

If you have levels equations for which the natural data is not a solution, you can ask the TABLO to add homotopy terms. This can make it relatively easy to obtain results which are a solution to these levels equations.

HOMOTOPY is now a reserved word (see section 4.2.1 of GPD-2).

You can ask TABLO to automatically add HOMOTOPY terms to all or selected levels equations.

1. This can assist in obtaining solutions to levels equations for which the natural data is not a solution. For example, this can be used to produce (eg, from data for one year) a non-steady state intertemporal data base which satisfies all the equations of the model, including the intertemporal ones (which probably would not be satisfied if the data for this one year is replicated to all years).
2. This can be used to find (based on an initial guess which is not a solution) an accurate solution to a system of levels equations.

We give examples below which illustrate these points.

The methods described here and automated by TABLO are methods introduced to the CoPS/IMPACT group of modellers by Mark Horridge. They have their origins in the so-called homotopy methods of mathematics (see, for example, the survey article by Watson (1986)) which can be used in various branches of science and engineering. In particular these methods are now an integral part of the MONASH (and other related) models [Dixon and Rimmer (2002)]. These methods are very powerful, as the examples below indicate.

Unlike Newton's method (see sections 7.5 and 7.5.4), this methodology will work when only some of the equations of the model are written as levels equations in the TAB file. [There may also be many linear equations written in the TAB file.] Also, even if all Equations are Levels Equations, the solution may converge using the adding Homotopy terms method in cases where it would not converge using Newton's method starting from the same initial guesses.[117]

7.6.1 Example 1 – Solving a System of Levels Equations

Consider the two levels equations

$$Y = X + a$$
$$X^2 + Y^2 = 5$$

where "a" is a constant (that is, a parameter) and X,Y are the unknowns. We show how adding HOMOTOPY terms to (one of) the levels equations makes it easy to obtain accurate solutions of this system of (levels) equations. A suitable TAB file is shown below.

```
!  Set defaults for Levels model   !
EQUATION(DEFAULT=LEVELS) ;
VARIABLE(DEFAULT=LEVELS) ;
FORMULA(DEFAULT=INITIAL) ;
COEFFICIENT(DEFAULT=PARAMETER) ;
VARIABLE(DEFAULT=CHANGE) ;

Set COM (c1-c3) ;

Variable (all,c,COM) V1(c) ;
Variable (all,c,COM) V2(c) ;

Coefficient (parameter) (all,c,COM) coeff1(c) ;
Formula (Initial) (all,c,COM) coeff1(c) = $POS(c) ;

Formula (Initial) (all,c,COM) V1(c)= $POS(c)+2 ;
Formula & Equation eq1 (all,c,COM) V2(c) = V1(c) + coeff1(c) ;

Equation (ADD_HOMOTOPY) eq2 (all,c,COM) V1(c)^2 + V2(c)^2 = 5 ;
```

[117] The automatic adding of homotopy terms to levels equations was introduced in Release 8.0.

Here the single system of two equations above has been replaced by three different systems (one for each "c" in the set COM). The parameter "a" has been replaced by "coeff1(c)". The unknowns X and Y have been replaced by V1(c) and V2(c) respectively. Note that we have included the statement "Variable(Default=Change)" to make all associated linear variables changes variables. This is because the variables in question could be positive, negative or zero.

The assignment of initial (that is, pre-simulation) values to the variables is important and interesting. Here we have assigned essentially arbitrary initial values for V1(c) [V1(c)=$POS(c)+2]. Then those for V2(c) are naturally given by the Formula & Equation eq1 (which makes this equation satisfied by these initial values). Of course it is highly likely that the second equation eq2 will NOT be satisfied by these initial values.

This is why we have asked TABLO to add the HOMOTOPY terms to this equation eq2 via the "(ADD_HOMOTOPY)" qualifier in that equation. TABLO adds a term

CX(c)*HOMOTOPY

to the right-hand side of the equation.[118] Here HOMOTOPY is a Variable(Levels,Change) whose initial value is –1 and CX(c) is a Coefficient(Parameter) whose value is calculated so as to make the equation satisfied with the given initial values for V1(c) and V2(c). [Consider the "c1" equation. The initial values of V1 and V2 are V1("c1")=3 and V2("c1)=V1("c1")+COEFF1("c1")=3+1=4. Thus the LHS of eq2 is equal to $3^2 + 4^2 = 25$ while the RHS is equal to 5+CX("c1")*(-1) since the initial value of HOMOTOPY is –1. Thus the initial value of CX("c1") must be –20 to make the two sides equal.]

To solve these equations, we carry out a simulation in which only the HOMOTOPY variable is shocked (by 1, from its initial value of –1 to its final value of 0). Notice that when HOMOTOPY has the value zero, the original equation eq2 (without the CX(c)+HOMOTOPY term) is satisfied. This is why the post-simulation values of the endogenous variables V1(c) and V2(c) are solutions of the two equations in question.

If you carry out the simulation you will find that the linear results are c_V1("c1")=c_V2("c2")=–2. This means that the solution to the "c1" versions of the equations is V1("c1)=1 and V2("c2")=2 (since the c_V simulation values are changes from the initial values of 3 and 4 respectively).

This is a simple example showing how adding HOMOTOPY terms to selected levels equations can make it easy to use GEMPACK to find solutions of a system of levels equations (starting from an initial guess – namely the pre-simulation values assigned by the READ and FORMULA(INITIAL) statements).

Note that HOMOTOPY terms only need to be added to selected equations. Note also that, in the example above, the other equation eq1 could have been included as a linearized equation in the TAB file without affecting the procedure.

You can find a similar example in section 7.8.2. There the HOMOTOPY terms are added explicitly in the TAB file. That section contains an explanation as to why the solution obtained satisfies the original levels equations.

7.6.2 Example 2 – Finding an Intertemporal Data Set for an Intertemporal Model

A data base for an intertemporal model consists of an input-output table for each year represented in the model. Since the model typically goes out into the future, input-output tables for these years are not available and have to be made up. Often this is done by taking the most recently available input-output table (say for the year 2000) and then replicating it for the other years (into the future). The problem is that the resulting intertemporal data set is almost certainly not a solution to the levels equations of the model. It will provide a solution to the "intra-period" equations (those involving just a single time instant) but will almost certainly not be a solution to the "inter-period" equations (those involving two or more time instants).

Of course it is important to be able to obtain an intertemporal data set which provides a solution of all the equations (including the inter-period equations) since a simulation in GEMPACK must start from such a solution. [GEMPACK simulations move from one solution to another. They produce unreliable results if they do not start from such an initial solution – see section 7.8 for more details.]

This problem of finding an intertemporal data set is a well known (and serious) problem for modellers building and solving intertemporal models using GEMPACK (see, for example, Codsi *et al* (1992) and Wendner (1999)).

---

[118] See section 7.6.4 below for formal details about the ADD_HOMOTOPY qualifier and the statements it generates. In the example in the text, the Coefficient CX(c) will actually be called eq1@H(c).

Adding Homotopy terms to the inter-period equations and then carrying out a simulation in which the Homotopy variable is shocked can solve this problem for many intertemporal models. See section 7.6 of GPD-2 for more details and a detailed worked example.

7.6.3 Example 3 – Use When Adding Behaviour to a Model

Sometimes when you add behaviour to a model, there is not an obvious solution of the underlying levels equations you wish to add. For example, this happens in several parts of the MONASH model [Dixon and Rimmer (2002)]. Explicit homotopy terms were added to equations by Peter Dixon, Maureen Rimmer and colleagues to get around the fact that the pre-simulation data does not give an initial solution. In MONASH.TAB [see chapter 4 of Dixon and Rimmer (2001)], the linearized homotopy variable is called **del_unity**. Examples of equations using del_unity include E_del_f_ac_p_y, E_d_f_fea_t_j, E_d_f_fcfc_t**.**

Considerable technical skill is required to add the correct homotopy terms in these cases by hand. Now it is easy since you can ask TABLO to do it for you. All you have to do is to

- add the relevant equation as a levels equation,
- tell TABLO to add the appropriate HOMOTOPY terms by including the qualifier "(ADD_HOMOTOPY)" when you declare the equation.

Of course adding levels equations does require a little technical skill. Help is available in the document Harrison and Pearson (2002). The relevant equations are naturally levels equations so inserting them as such in the TAB file is a sensible thing to do.

If you choose this route, you just need to remember to shock the linear variable c_HOMOTOPY by one in simulations.

Example from MONASH

The equation E_del_f_ac_p_y adjusts the quantity of capital from the pre-simulation value QCAPATT read in from the database to the new value QCAPPLUS1_B which is the quantity of capital after subtracting the depreciation and adding in the investment.

```
Formula(initial) (all,j,IND) QCAPPLUS1_B(j)=QCAPATT(j)*(1 - DEP(j)) + QINVEST(j)
```

The natural levels equation, which sets QCAPATT equal to the parameter QCAPPLUS1_B, is

```
Equation(Levels)  E_del_f_ac_p_y

(All,j,IND) QCAPATT(j) = QCAPPLUS1_B(j) + F_AC_P_Y(j) ;
```

where F_AC_P_Y(j) is a shifter which is initially equal to zero. (It is used to turn the equation off if we do not wish to use the equation.)

However this levels equation is not satisfied initially since QCAPATT on the data base is not equal to QCAPPLUS1_B. [The value of QINVEST(j) is effectively also read from the data base. The pre-simulation values of QCAPATT and QCAPPLUS1_B are only equal in the unlikely case that investment QINVEST in the pervious year exactly offsets depreciation.]

Below we show how this equation E_del_f_ac_p_y could be added to MONASH.TAB as a levels equation with the ADD_HOMOTOPY qualifier. The equation is:

```
Equation(Levels, ADD_HOMOTOPY) E_del_f_ac_p_y

(All,j,IND) QCAPATT(j) = QCAPPLUS1_B(j) + F_AC_P_Y(j) ;
```

In order to obey the rules for levels equations, terms in this equation must be either levels variables or Coefficient(Parameter)s – see section 3.9 of GPD-2. QCAPATT needs to be declared as a levels variable with associated linear variable cap_at_t, and QCAPPLUS1_B needs to be declared as a Coefficient(Parameter).

```
Variable(levels, linear_name=cap_at_t)  (All,j,IND)
QCAPATT(j) # Quantity of capital stocks, start of forecast year, ie K(t) Fq#;

Coefficient(Parameter) (All,j,IND)
QCAPPLUS1_B(j) # Qty of cap stock, end of data year, base  FIq#;

Formula(Initial) (All,j,IND) QCAPATT(j) = VCAP_AT_T(j)/PCAP_AT_T(j);

![[!Variable  (All,j,IND) cap_at_t(j)
# Capital stock at t (start of forecast year) #; !]]!
```

[See sections 3.4 and 2.2.2 of GPD-2 for information about the Variable qualifiers "linear_name=" used above and "linear_var=" used a couple of lines below.]

We need to introduce the levels variable associated with the linear variable del_f_ac_p_y and set it initially to zero.

```
Variable(levels,linear_var=del_f_ac_p_y)(All,j,IND) F_AC_P_Y(j);

Formula(initial) (all,j,IND) F_AC_P_Y(j) = 0.0 ;
```

When the homotopy equation is linearized, it is replaced by[119]

```
Formula(Initial)  !for new Coefficient(Parameter) CHOMO1@H!
(All,j,IND) CHOMO1@H(j) = QCAPATT(j) - {QCAPPLUS1_B(j) + F_AC_P_Y(j)} ;

Equation(linear) E_del_f_ac_p_y     (all,j,IND)
QCAPATT(j)/100.0 * cap_at_t(j)=del_f_ac_p_y(j) - CHOMO1@H(j)*c_HOMOTOPY;
```

The coefficient of c_HOMOTOPY gives the homotopy correction term which is calculated in terms of **initial** or base values as

```
[QCAPATT(j) - {QCAPPLUS1_B(j) + F_AC_P_Y(j)} ]
```

The original equation in the MONASH.TAB in Dixon and Rimmer (2001) is the linear equation:

```
Equation E_del_f_ac_p_y
  # Gives shock in yr-to-yr forecasting to capital at beginning of year t #
(All,j,IND) [QCAPATT(j) + TINY]*cap_at_t(j)
= 100*{QINV_BASE(j) - DEP(j)*QCAPATT_B(j)}*del_unity + 100*del_f_ac_p_y(j);
```

However since

```
[QINV_BASE(j) - DEP(j)*QCAPATT_B(j)}] = - QCAPATT(j) + QCAPPLUS1_B(j)
```

and F_AC_P_Y(j)} is initially zero, the term for the variable del_unity in the original equation in MONASH.TAB is the same as the term for c_HOMOTOPY so that the linearized homotopy equation is the same as the original equation in MONASH.TAB.

In order to run a simulation with the levels version of this equation, you would need to add the variable c_HOMOTOPY to the exogenous list and shock it by 1. Add to your Command file the statements:

```
exogenous c_homotopy ;
shock c_HOMOTOPY= 1.0;
```

7.6.4 Formal Documentation about ADD_HOMOTOPY

Here we describe in more formal terms what TABLO does whenever the qualifier "(ADD_HOMOTOPY)" occurs in a levels equation.

We illustrate this with the example equation:

```
Equation (Levels, Add_Homotopy) EQ1 (all,c,COM) X1(c) = A*X2(c) ;
```

where X1 and X2 are levels variables and A is a parameter. With this equation, TABLO will

- add a new Coefficient(Parameter) (All,c,COM) EQ1@H(c) [the name is based on the Equation name EQ1],
- give EQ1@H initial values via the Formula(Initial) (all,c,COM) EQ1@H(c) = X1(c) – A*X2(c),
- add the term –EQ1@H*HOMOTOPY to the RHS of equation EQ1 so that it reads (All,c,COM) X1(c) = A*X2(c) – EQ1@H(c)*HOMOTOPY,
- linearize this modified equation in the usual way.

In general, TABLO

- adds a new Coefficient(Parameter) whose name has "@H" added to the name of the equation,
- adds a Formula(Initial) for this Coefficient, and

[119] See section 7.6.4 below for formal details about the ADD_HOMOTOPY qualifier and the statements it generates.

- adds the term –<coef>*HOMOTOPY to the RHS of the levels equation.[120]

The linearized equation is shown on the INF (Information) file in the usual way (see section 9.3 of GPD-2). Just above it on the INF file you will see the Formula(Initial) and the name of the new Coefficient. In the example above, you will see the following on the INF file.

```
!  FORMULA(INITIAL) for new Coefficient(Parameter) EQ1@H
   (ALL,c,com) EQ1@H(c) = x1(c) - {A * x2(c)} ;  !

!  Add term -EQ1@H*HOMOTOPY to RHS of levels eq EQ1 !
!  EQUATION(LINEAR) EQ1
 (ALL,c,com) x1(c) * p_x1(c) = A * x2(c) * p_x2(c) - EQ1@H(c) * 100.0
        * c_HOMOTOPY ;  !
```

The levels variable is **always** called HOMOTOPY. It is always a (Levels,Change) variable whose initial value is set to –1. You should always shock its associated linear variable **c_HOMOTOPY** by 1 (which takes HOMOTOPY from –1 to 0).[121]

The Coefficient added is a perfectly normal Coefficient. For example you can look at its values via xwrite statements (even WRITE statements after the equation in the TAB file) and in AnalyseGE.

7.6.5 ADD_HOMOTOPY and NOT_ADD_HOMOTOPY Default Statements

Any Levels Equation can be given one of these qualifiers **ADD_HOMOTOPY** or **NOT_ADD_HOMOTOPY**.[122] If so this tells TABLO whether or not to add Homotopy terms to this equation.

There are also DEFAULT statements (see section 3.20 of GPD-2)

```
EQUATION ( DEFAULT = ADD_HOMOTOPY) ;
EQUATION ( DEFAULT = NOT_ADD_HOMOTOPY) ;
```

which you can place anywhere in a TABLO Input file. After the first of these Default statements, Homotopy terms will be added to all Levels Equations encountered (unless the Equation has its own ADD_HOMOTOPY or NOT_ADD_HOMOTOPY qualifier). After the second of these Default statements, Homotopy terms are not added. You can use these Default statements to turn on or off the adding of Homotopy terms to groups of Levels Equations in your TAB file. These Default statements have no effect on Linear Equations.

7.7 Options for Saving Solution and Updated Data Files

7.7.1 Saving Solutions or Updated Data After Separate Multi-step Calculations

When you carry out two or three separate multi-step calculations and then extrapolate from them, you normally do not see the solutions or the updated data after each separate multi-step calculation. [You just see the extrapolated solution and the data updated using that solution.]

However, it is possible to also save the save the solutions after each multi-step calculation and/or to save the updated data after each multi-step solution. There are Command file statements which allow these.

---

[120] If the equation name has more than 10 characters (so that adding "@H" would make it longer than the allowed Coefficient name length), TABLO makes up names of the form CHOMOddd where "ddd" are digits added to make the name a new one (for example, CHOMO5 or CHOMO143). If the equation EQ1 only has one side (for example B=0 or 0=B), the homotopy term is added on that side, the Formula(Initial) says EQ1@H=B, and the term added to the levels equation is +EQ1@H*HOMOTOPY (assuming that the equation has name "EQ1").

[121] If you start from a data base which does provide a solution to the underlying levels equations (that is, you can be sure that all levels homotopy correction values are zero based on this data), you do not need to shock c_HOMOTOPY. But even then there is no harm in doing so since the variable is multiplied by zero so the shock will do nothing.

[122] These qualifiers must not be added to Linear Equations.

You can use the following statements in your Command file to save one or all of these solutions.

SSL = all ; ! to save all multi-step solutions on the solution file

SSL = last ; ! to save the last multi-step solution

SSL = none ; ! to save just the cumulative solution - the default.

These extra solutions are saved as "subtotals" so when running GEMPIE you can print them out as columns next to the usual cumulative solution (see section 10.1.3). You can also access them via ViewSOL.

> For example, if you were running 1, 2, 4 Euler solution for your simulation, and you add to your command file:
>
> SSL = all ;
>
> the three separate solutions corresponding to a 1-step Euler, a 2-step Euler and a 4-step Euler will be written on the Solution file as well as the extrapolated solution calculated from these three separate solutions. When you run GEMPIE you can choose to print them out.
>
> For SSL= last ; you would just save the 4-step Euler and the extrapolated solutions.

To save the updated data files, after the two or three multi-step solutions, the syntax is similar:

SUP = all ; ! to save the updated data after each multi-step

SUP = last ; ! to save after the last multi-step

SUP = none ; ! the default.

The files names have the same file stem but use suffixes .UD5,.UD6,.UD7 (cf. section 4.5). For option SUP, the data files should be Header Array files. For original data files which are text files, the UD5, UD6 and UD7 files are saved in Header Array format (which can be looked at using ViewHAR or SEEHAR).

Neither of these options can be used if you have more than one subinterval (see section 7.3 above) or with user-specified accuracy (see section 7.4 above).

7.7.2 Saving Updated Values from All FORMULA(INITIAL)s

It is possible to ask GEMSIM or a TABLO-generated program to save updated values of all coefficients whose initial values are set via FORMULA(INITIAL)s. To do this, put statements

```
sui = yes ;
updated formula initial data = <filename> ;
```

in your Command file.[123] Then updated values of all coefficients (or parts of coefficients) whose initial values are set via a FORMULA(INITIAL) which does not have "WRITE UPDATED VALUE .." (see section 4.11.8 in GPD-2) in it will be written to this file.

Example

If you add the following statements to the Command file SJLB.CMF (see Figure 2.8.1 in GPD-1)

```
sui = yes ;
updated formula initial data = sjlbfi.upd ;
```

the updated data file SJLBFI.UPD will contain the post-simulation values of prices and quantities initialised via FORMULA(INITIAL)s in SJ.TAB.

[123] Alternatively, select option **SUI** from the opening options screen (see chapter 14). [This is equivalent to the statement "sui = yes ;" in a Command file.]

7.8 Solutions Report Perturbations of the Initial Values

When you solve a model using GEMPACK, the results reported (percentage changes or changes) are perturbations from the original values of the underlying levels variables. In particular, if the pre-simulation values of the Coefficients corresponding to the levels variables do not satisfy the levels equations of the model, the implied post-simulation values will not satisfy the equations. We give examples to illustrate this important point. In section 7.8.2 below, we show how the introduction of Homotopy terms (see section 7.6) can be used to produce post-simulation values which do satisfy the desired levels equations (even when the pre-simulation values do not).

Example 1.

Consider the very simple model with just one levels equation

$$X = 2Y \qquad \text{(E1.1)}$$

The natural linearized equation to use is

$$p_X = p_Y \qquad \text{(E1.2)}$$

which is obtained from the Product rule (see chapter 9 of GPD-2) since 2 is a constant.

Suppose that you start a simulation from a point where X=1 and Y=1. Of course this is not a solution of the underlying levels equation (E1.1). If, say, Y is exogenous and is shocked by 10 percent, clearly from (E1.2) the simulation result for p_X will also be 10 percent. This would suggest that the post-simulation point is given by X=1.1 (10 percent more than 1) and Y=1.1. This point is not a solution of the underlying levels equation (E1.1) either.

Alternatively, you could use the Change Differentiation form

$$c_X = 2*c_Y \qquad \text{(E1.3)}$$

Or, equivalently,

$$X*p_X = 2*Y*p_Y \qquad \text{(E1.4)}$$

A TAB file for this, starting from X=1=Y is as follows.

```
Variable p_X ; Variable p_Y ;
Coefficient X ; Coefficient Y ;
Formula (Initial) X = 1 ;  Formula (Initial) Y = 1 ;
Update X = p_X ; Update Y = p_Y ;
Equation eq1  X*p_X = 2*Y*p_Y ;
```

If you solve this (say Gragg 2,4,6) giving a 10 percent shock to Y, you would find that the result for p_X is 20. This would suggest the post-simulation point X=1.2 (20 percent more than 1), Y=1.1.

Why do these two linearizations give such different results? There are several things to consider.

1. Firstly, since the starting point in each case (X=Y=1) does not satisfy the underlying levels equation (E1.1), there is nothing in the theory underpinning the GEMPACK solution methods to suggest that you should end up at solution.
2. Secondly, note that the derivation of the Product Rule (see the Example in section 9.2.1 of GPD-2) specifically relies on the fact that the simulation starts from a solution to the underlying levels equation (since R=CPQ is substituted into the equation (9) there which is obtained by Change differentiation). Thus a the linearization (E1.2) is particularly inappropriate when the starting point is not a solution of the underlying levels equation.
3. Thirdly, note that the derivation of the Change Differentiation rules (see section 9.2.1 of GPD-2) does not appear to rely on substituting back the original levels equation. Thus the use of the change linearized form (E1.4) above does not seem to be quite so suspect. We elaborate on this point in section 7.8.1 below.

7.8.1 Change Differentiation Keeps a Constant Error

It turns out that, if you only use Change Differentiation (that is, avoid the use of any Percentage-change rules – see section 9.2.2 of GPD-2), the "solution" that GEMPACK reports has the same error in the underlying levels equations as at the initial point.

To see what we mean by the "same error" consider again Example 1 above and consider the results based on the Change differentiation version (E1.4). The starting point is X=Y=1. The error in the underlying levels equation (E1.1) is RHS–LHS=2–1=1. The "solution" gives X=1.2, Y=1.1. The error at this point is

RHS – LHS = 2*1.1 – 1.2 = 1 .

Hence the error is the same.

This illustrates the general point that

If only Change Differentiation is used,
the difference between the post-simulation LHS and RHS values
is the same as that between their pre-simulation values.

In particular, when the pre-simulation values satisfy the underlying levels equations, so do the implied post-simulation levels values (since the difference is zero in both cases).

To understand the general point highlighted above, recall that GEMPACK uses only the linearized equations when it solves the model. But the linearized equations are no different if we add a constant to the either side of any of the levels equations. If, in Example 1 above, we had been thinking of the levels equation

$$X + 1 = 2Y \qquad \text{(E1.5)}$$

(instead of X=2Y), this equation would be satisfied by the pre-simulation values X=1,Y=1. The Change Differentiation of this equation (E1.5) is the same as (E1.4). This levels equation (E1.5) is also satisfied by the implied post-simulation values X=1.2,Y=1.1.

Because GEMPACK only uses the linearized equations, when it solves the model it is in some sense integrating these linearized equations. As you know from elementary calculus, integration only determines the result up to an arbitrary constant.[124] Extra information is needed to determine that constant. In the GEMPACK case, this constant is the difference between the LHS and RHS when the pre-simulation values of the relevant Coefficients are substituted into the levels equation.

We give a second, less trivial example of what happens if the starting point does not satisfy the underlying levels equation and where only Change Differentiation is used.

Example 2.

Consider the model with just one levels equation

$$X^2 + Y^2 = 3XY \qquad \text{(E2.1)}$$

The TAB file is shown below.

```
Variable (Levels) X ; Variable (Levels) Y ;
Formula (Initial) X = 2 ;  Formula (Initial) Y = 1 ;
Equation (Levels) eq1  X^2 + Y^2 = 3*X*Y ;
```

The pre-simulation levels values X=2,Y=1 do NOT satisfy the equation (since the LHS=5 and the RHS=6).

Suppose that you solve this model, taking Y as exogenous and X as endogenous, and give a shock so that Y goes from its pre-simulation value of 1 to a post-simulation value of 1.6. The simulation results (using Gragg 20,30,40 steps to get accurate results) show a percentage change of 94.1619 for X. This increases X from 2 to 3.88324.

Do the post-simulation values of X and Y satisfy the levels equation eq1? The answer is clearly NO since the LHS=17.63955 while the RHS=18.63955. The RHS is still 1 more than the LHS (as it was with the pre-simulation values).
Note, however, that the error (difference between RHS and LHS values) is the same at the post-

[124] For example, $\int x\,dx = x^2/2 + C$.

simulation point as it is at the pre-simulation point. This a further example of the general point highlighted above.[125] [TABLO uses Change differentiation when differentiating equation (E2.1) because of the + sign on the LHS – see section 2.2.5 of GPD-2.[126]]

7.8.2 How Can Homotopies Help?

Suppose that you have a levels equation you wish to include in your model but that you cannot easily obtain pre-simulation values of the relevant Coefficients which satisfy this equation. That might be the case for the equation (E2.1) in the example above. It is certainly the case in recursive dynamic models and intertemporal models for the capital accumulation equation if investment in the single year data base you are starting from is more or less than that required just to offset capital depreciation. [See section 7.6 above and section 7.6 of GPD-2.]

Introducing a Homotopy variable (this is a levels variable) and giving it a suitable pre-simulation value is a cunning way of ensuring that the post-simulation values do satisfy the desired levels equation, even though the pre-simulation values may not satisfy the equation.

The previous sentence may seem mysterious to you, and possibly it seems to contradict what we said in the section above. We explain more below.

Consider again the equation

$$X^2 + Y^2 = 3XY \qquad \text{(E2.1)}$$

from Example 2 in section 7.8.1 above. Suppose that this is the equation we wish to include in our model. Suppose, as in the example above, we start with pre-simulation values X=2,Y=1. [These do not satisfy the equation.] How can we achieve post-simulation values which satisfy this equation? The secret is to **modify** the equation in the TAB file by adding a suitable Homotopy term. The modified equation is

$$X^2 + Y^2 = 3XY + C*HOMOTOPY$$

where HOMOTOPY is a levels, change variable, and C is a suitable constant.

Suppose that we take the pre-simulation levels value of HOMOTOPY to be –1 and choose the value of C so that this modified equation is satisfied with these pre-simulation values, namely

$$X=2,\ Y=1,\ HOMOTOPY= -1.$$

Clearly we need C=1. Thus the modified equation is

$$X^2 + Y^2 = 3XY + HOMOTOPY \qquad \text{(E2.2)}$$

Now, when we solve the model, as well as shocking Y by 60 percent, we shock HOMOTOPY from its pre-simulation value of –1 to a post-simulation value of 0 (that is, give a shock of +1 to c_HOMOTOPY). Since the pre-simulation values (of X,Y,HOMOTOPY) satisfy the levels equation (E2.2), so will the post-simulation values. But the post-simulation value of HOMOTOPY is zero. Hence the post-simulation values for X and Y must satisfy the original equation (E2.1).

If you carry out this simulation after modifying the TAB file from the section above to include the statements[127]

```
Variable (Levels, Change) HOMOTOPY2 ;
Formula (Initial) HOMOTOPY2 =  -1 ;
Equation (Levels) eq1  X^2 + Y^2 = 3*X*Y + HOMOTOPY2 ;
```

[125] Here, if we had used the levels equation $X^2 + Y^2 = 3XY - 1$ in the TAB file for Example 2 [adding the constant "–1" to the RHS], the Change linearization would be the same, this equation would be satisfied by the pre-simulation values X=2,Y=1 and it is also satisfied by the post-simulation values X=3.88324,Y=1.6.

[126] If you wanted to check this, you could look in the Information file to see the linearized version (see section 9.3 of GPD-2). Or you could use TABLO option ACD (see section 2.2.6 of GPD-2) to ensure that it happens.

[127] We use HOMOTOPY2 instead of HOMOTOPY since the latter is a reserved word which is used with the ADD_HOMOTOPY equation qualifier (see section 7.6).

you will find that the simulation result for p_X is 109.4425 which means that the post-simulation levels value of X is 4.18885.[128] Note that X=4.18885, Y=1.6 do satisfy the desired equation (E2.1) since LHS=20.1065=RHS.

This method relies on

- adding a suitable HOMOTOPY term to the levels equation (in the TAB file),
- taking the pre-simulation value of HOMOTOPY to be –1,
- calibrating the constant which multiplies the HOMOTOPY variable so that, when the pre-simulation values of the other Coefficients and of HOMOTOPY are substituted in, the modified equation is satisfied,
- shocking c_HOMOTOPY by +1 so that is post-simulation value is 0.

When this is done, the implied post-simulation values must satisfy the original (desired) levels equation since the post-simulation value of the term added is zero.

Note that this method does not contradict the statement

"the difference between the post-simulation LHS and RHS values is the same as that between their pre-simulation values"

in the section above. Rather this method exploits the truth of this statement. The equation in the TAB file is the modified one. Only because of a cunning choice of the post-simulation value of HOMOTOPY (namely, zero) does it follow that the desired equation is satisfied (because the modified one must be satisfied, and the modified one is identical to the desired one when the HOMOTOPY variable takes the value zero).

Note that, by just shocking c_HOMOTOPY by 1 and giving no shock to the other exogenous variables, you can use this method to modify the pre-simulation values of the endogenous variables so that they do satisfy the desired equation.

In the example above, if you give a zero shock to p_Y and still shock c_HOMOTOPY by 1, you find the p_X result is 30.9017 which means that the post-simulation value for X is 2.61803. These values X=2.61803, Y=1 satisfy the desired equation (E2.1) since LHS=7.85408 while RHS=7.85409. This is a way of obtaining a value for X which satisfies the desired equation (E2.1) when Y=1.

Note that, when you use the ADD_HOMOTOPY qualifier introduced in section 7.6 above, TABLO writes statements which automatically work out the desired value of the constant C [see equation (E2.2) above]. This Coefficient C is the parameter whose name has "@H" added to the end of the name of the equation in the TAB file (see section 7.6.4 above).

[128] Again use Gragg 20,30,40 steps in order to obtain accurate results.

CHAPTER 8

8. Solution Files

Solution files contain the results of a simulation (that is, the changes or percentage changes in the endogenous variables). They also contain details of the shocks and closure.

In section 8.1 we summarise the Command file statements relating to Solution files. In section 8.2 we describe the contents of a Solution file in a little detail, and we summarise the availability of levels results in section 8.3. We introduce Solution Coefficient (SLC) files and variants of these in sections 8.4 to 8.7.

8.1 Command File Statements Related to Solution Files

- The name of the Solution file name can be specified via a statement of the form

  Solution file = <file_name> ;

  where <file_name> is the name of the Solution file to be created with the suffix omitted. The standard Solution suffix .SL4 will be added by the program.

  However, for the reasons set out in section 2.5, we recommend that you **do not include such a statement** (provided that your Command file has the usual suffix .CMF). Then the Solution file is given the same name as the Command file stem, though with the suffix .SL4 (see section 2.5.1).

- Command files used to run simulations must have a Verbal Description statement:

  verbal description = <brief description of the simulation > **;**

  where the description may be one or more lines and the last line must end with a semicolon. Each line must contain no more than 79 characters.

  For example, the verbal description in the Command file SJLB.CMF (see section 2.8 of GPD-1) is:

  ```
  verbal description =
  Stylized Johansen model. Standard data and closure.
  10 per cent increase in amount of labor.
  (Capital remains unchanged.) ;
  ```

  GEMSIM and TABLO-generated programs automatically append the solution method, number of steps and number of subintervals (if more than 1) to the Verbal Description of the simulation. [If you use a Command file to carry out a simulation, its name is also added to the Verbal Description (see section 2.5.1).]

- Usually the Solution file contains the results for all endogenous variables. However it is possible to select a list of endogenous variables to be retained on the Solution file using the statement:

  cumulatively-retained endogenous <list> ;

  where <list> is a list of variables to be retained using the syntax in section 5.4.

- GEMSIM, TABLO-generated programs and SAGEM can calculate subtotals results and store them on the Solution file. The Command file statement uses the syntax

  subtotal <list> = <description> ;

where <list> is a list of shocked variables to sum over. See chapters 10 and 11 for more details about subtotals.[129]

- If your TABLO Input file includes levels variables or reports levels variables using the ORIG_LEVEL technique in section 4.5.5 of GPD-2, your Solution file will also contain levels results. However levels results can be turned off using the command:

  levels results = no ; ! the default is YES

8.1.1 SAGEM Only Command File Statements

In a SAGEM Command file you can also specify the *individual column* results (see section 10.1.1) using:

```
individually-retained exogenous <list> ;
individually-retained endogenous <list> ;
```

where <list> is a list of variables to be retained using the syntax in section 5.4.

8.2 Contents of Solution Files

A Solution file contains the following information in binary form. The various utility programs such as GEMPIE, SLTOHT and ViewSOL can then read some of this information and present it to you in a form you can read or print.

- The variable names and set and element names, # labelling information (see section 4.2.2 of GPD-2) for variables and sets. This can then appear on Print files (PI5) produced by GEMPIE and other output files created by SLTOHT. This information is also shown by ViewSOL.
- The time and date of the simulation run when the Solution file was created.
- Which TABLO-generated program or GEMSIM version was used to run the simulation.
- The TABLO Input file and condensation STI file (if used) – see section 8.2.1.
- The Command file for the simulation – see section 8.2.2.
- The verbal description – see section 8.1.
- The closure (which can be recovered using the program SEENV – see chapter 12 of GPD-4).
- The shocks.
- The endogenous results (sometimes called the **cumulative** or **totals solution**). Usually the results for all endogenous variables are available on the Solution file but only some of these will be available if a "cumulatively-retained endogenous = <list> ;" statement was included in the Command file.
- Levels results, if available (GEMSIM and TABLO-generated programs only, see section 8.3).
- Subtotal results, descriptions and a list of the shocks producing each subtotal (see chapters 10 and 11).
- Individual column solutions (SAGEM only) when the simulation has solved for individual column results – see chapter 10.

Note that Solution files produced by GEMSIM and TABLO-generated programs

- always contains a totals column (that is, the cumulative or totals results),
- may contain levels and/or subtotals results,
- never contain individual column results.

On the other hand, Solution files produced by SAGEM

[129] Subtotals were introduced for GEMSIM and TABLO-generated programs in Release 7.0. Subtotals have been available via SAGEM and GEMPIE since 1990.

- may contain a totals column (that is, the cumulative or totals results),
- never contain levels results,
- may contains subtotals results,
- may contain individual column results.

Both GEMPIE and ViewSOL can access the cumulative solution and any levels or subtotals results available on the Solution file. See chapter 7 of GPD-4 for details about selecting these in the case of GEMPIE. For ViewSOL, whether or not you see the levels results or subtotals results depends on the settings under the ***File*** **|** ***Options*** menu.

GEMPIE can access the individual column results produced by SAGEM (see chapter 7 of GPD-4) but ViewSOL cannot be used to access these results.

8.2.1 TABLO File and TABLO STI File Stored on Solution File

In Release 6.0 (or later), the TABLO Input file is stored on the Auxiliary Table (.AXT or .GST) file (unless option NTX was selected when you ran TABLO). [See section 2.1.1 of GPD-2.]

In Release 7.0 (or later), this TABLO Input file is transferred from the Auxiliary Table file (providing it is there) to the Solution file when you carry out a simulation using GEMSIM or a TABLO-generated program. This means that you can use the program TEXTBI (see chapter 14 of GPD-4) to recover that TABLO Input file from the Solution file. This may assist in understanding simulation results.

Similarly, if you use a Stored-input file to run TABLO, this Stored-input file is now transferred to the Auxiliary Table file produced by TABLO (unless TABLO option NTX is selected). This Stored-input file is also transferred to the Solution file when you run a simulation using GEMSIM or a TABLO-generated program. You can use TEXTBI to recover the Stored-input file from the Solution file or from the Auxiliary Table file. [Strictly speaking, the Stored-input file put on the Auxiliary Table file is the one used for the CODE stage of TABLO. If you stopped and restarted TABLO – see section 2.5 of GPD-2 – condensation actions may not be on the Stored-input file put on the Auxiliary Table or Solution file.]

This means that it is usually possible to recover the TABLO file and any condensation actions from any Solution file.

Since the original data is stored on the SLC file (see section 8.4), this means that you can recover everything about a simulation from the Solution and SLC files.

8.2.2 Command File Stored on Solution file

The Command file for the simulation is stored on the Solution file. This means that you can use the program TEXTBI (see chapter 14 of GPD-4) to recover that Command file from the Solution file. This may assist in understanding simulation results.

8.3 Levels Results

It is possible to ask GEMSIM or a TABLO-generated program to calculate and report pre-simulation and post-simulation levels values as well as the usual percentage change results.[130] Details can be found in sections 4.5.5 and 4.5.6 of GPD-2.

Note that levels results are not produced if your simulation starts from existing Equations and BCV files (see section 9.2.1). This is because the program may not have available the pre-simulation levels values of all coefficients (since the BCV file only contains the values of those coefficients needed for the backsolve and update stages).

SAGEM cannot produce levels results since SAGEM does not know the values of all Coefficients (just those that happen to be present in the Equations).

[130] This is true even if your TABLO Input file has only linear variables and linearized equations. Reporting levels results was new in Release 6.0.

8.4 Solution Coefficients (SLC) Files

When TABLO-generated programs or GEMSIM carry out a simulation, by default they produce a **Solution Coefficient file**, with suffix **.SLC** which contains the **pre-simulation** values of all Coefficients defined in the TABLO Input file.[131] The SLC file has the same name as the Solution file, but with the different suffix .SLC.

This SLC file is a Header Array file. The long names on each array indicate the name of the Coefficient whose values are held at each header. Each array on the file corresponding to a Real Coefficient contains set and element labelling (see chapter 5 of GPD-4).[132]

This SLC file is designed for use with the Windows program **AnalyseGE** (see section 2.6 of GPD-4). This is aimed at assisting modellers to analyse their simulation results. AnalyseGE makes it easy for users to calculate expressions involving Coefficients and Linear variables such as those occurring in linearised equations. AnalyseGE obtains the values of variables from a Solution file and the values of Coefficients from the associated SLC file. See section 2.6 of GPD-4 for details of AnalyseGE and also its own Help file. See section 6.1 of GPD-8 for a hands-on introduction to AnalyseGE.

The same time and date stamp is put on the Solution and SLC files produced in a single run of a TABLO-generated program or GEMSIM. The program AnalyseGE uses these to alert the user if the files were not produced at the same time.

A related statement in Command files for GEMSIM and TABLO-generated programs is

SLC file = YES|no ; ! YES is the default. Include "slc file = no ;" to suppress the SLC file.

An SLC file cannot be created by GEMSIM or a TABLO-generated program if existing Equations and BCV files are being used (see section 9.2.1).

8.4.1 Solution and SLC Files are Complete Record of the Simulation

The Solution file and the Solution Coefficients file together contain a good record of the simulation carried out.

The SLC file contains the values of all the initial data for the simulation (since it contains the pre-simulation values of all Coefficients in the TABLO Input file). The Solution file itself contains the remaining information (including the closure, shocks, Command file, TABLO Input file and condensation STI file) required to re-run the simulation if necessary. So, together, the Solution and SLC files are a good audit trail for the simulation.

8.4.2 System-Initiated Coefficients on SLC File

If you condensed your model when running TABLO, you may wish to see the values of the system-initiated coefficients (the ones introduced by TABLO during condensation – see section 2.3.2 of GPD-2) in the SLC file. If so you can include the statement

```
SLC system_coefficients = yes ;
```

in your Command file. Then the values of all system-initiated coefficients will be put on the SLC file (and also on the UDC or AVC file if they are also being created – see section 8.5 below).[133]

To examine the values of a system-initiated coefficient in AnalyseGE, you can type its name (for example, C00134) into the Evaluate memo on the AnalyseGE form.

Of course you cannot see the formulas defining the system-initiated coefficients in the TABmate form of AnalyseGE. However, you could copy a formula from the INF or LOG file and paste it into the Evaluate memo on the AnalyseGE form. You could then edit the formula to calculate different parts of the right-hand side if you need to see how the values are built up.

---

[131] SLC files were introduced in Release 7.0.

[132] This set and element labelling was added to SLC files in Release 8.

[133] This was introduced in Release 8.0.

8.5 Updated and Average Coefficient Values (UDC and AVC) Files

Updated Coefficient Values (UDC) files are similar to Solution Coefficient Values (SLC) files (see section 8.4) except that they contain the **post-simulation values** of all Coefficients (whereas SLC files contain pre-simulation values).

Average Coefficient Values (AVC) files hold the average of pre-simulation values (as on an SLC file) and post-simulation values (as on an UDC file) for all Coefficients.[134]

Like SLC files, UDC and AVC files are Header Array files. The long names on the arrays indicate the names of the Coefficients whose values are held at each header. Each array on the file corresponding to a Real Coefficient contains set and element labelling (see chapter 5 of GPD-4).

These files have suffix **.UDC** and **.AVC** respectively.

Use the following statements in your Command file if you wish these files to be produced.

UDC file = yes|NO ; ! NO is the default

AVC file = yes|NO ; ! NO is the default

You can only ask GEMSIM or a TABLO-generated program to produce these files if you are also producing the SLC file. An AVC file cannot be produced unless both the SLC and UDC files are produced (since its values are the average of those on the SLC and UDC files). Like SLC files, UDC and AVC files are given time and date stamps so that AnalyseGE and other programs can tell if they were created in the same run as the Solution file with the same name.

Like SLC files, these UDC and AVC files can be used with AnalyseGE (see section 2.6 of GPD-4 and chapter 6 of GPD-8). When you load a Solution file into AnalyseGE and the UDC file or AVC file created at the same time as the Solution file is present, AnalyseGE allows you to use the Coefficient values on one of those files instead of those on the SLC file.

8.5.1 How Values on UDC Files are Calculated

When you ask for a UDC file to be created, the program (GEMSIM or a TABLO-generated program) carries out an extra pass in order to calculate values needed for the UDC file.

For example, suppose that you are doing an Euler 2,4,6-step calculation. Then, after the last step of the 2-step calculation, an extra pass 3 is done just to calculate the values of all Coefficients (that is, just the Reads/Formulas parts of this step are done). Similarly the Reads/Formulas are done in an extra pass 5 (after step 4 of the 4-step) and in an extra pass 7 (after step 6 of the 6-step). The values of the Coefficients calculated in these extra passes are the post-simulation values if that calculation (the 2-step, 4-step or the 6-step) were all that were being done. Then the values put on the UDC file are those extrapolated (in the same way that updated data is extrapolated) from the Coefficient values calculated in these extra passes.

If you are doing a Gragg 2,4,6-step calculation, then the solution is based on 3-pass, 5-pass and 7-pass calculations since an extra pass is done for Gragg (see section 12.2). In that case, extra passes (pass 4 after pass 3 of the 3-pass, pass 6 after the 5-pass and pass 8 after the 7-pass) for just the Reads/Formulas are made in which the values of the Coefficients are calculated. The values on the UDC file are extrapolated from these.

If you do only one or two multi-step calculations (for example just an Euler 4-step, or if you extrapolate from Gragg 2-step and 4-step results), again extra passes of the Reads/Formulas are done at the end of each multi-step calculation to calculate values of Coefficients. The values put on the UDC file are based on these (plus extrapolation if relevant).

8.5.2 Values on UDC Files May Differ From Values Calculated from the Updated Data

Values on the updated data can be read in as data for a new subsequent simulation. However if there are FORMULA statements setting some of the data values or recalculating some of the data, the pre-simulation values of some Coefficients may be different from what they were at the end of the previous simulation.

---

[134] UDC and AVC files were introduced in Release 8.

For example, some of the values of initial prices may always be set to one at the start of the simulation as a way of setting the volume units. However during a simulation, prices may change, and the values of quantities shown on the UDC file reflect this. Consider the simple example below.

Example.

```
Coefficient V # dollar value # ;  P # Price # ; X # Quantity # ;
Variable p_V ; p_P ; p_X ; ! percentage changes !
Read V from file … ;
Update V = p_P + p_X ;
Formula (Initial) P = 1 ;  Formula (Initial) X = V / P ;
Update P = p_P ; X = p_X ;
! plus various other equations involving p_X and p_P !
```

At the end of a simulation, the values of V, P and X may be 15, 1.5 and 10 respectively. These are the values which would appear on the UDC file. If the updated data from that simulation is read to begin a new simulation, the pre-sim values of V,P and X (that is, the values during the first step of that simulation) would be 15, 1 and 15 respectively because of the Formula(Initial)s in the TAB file. The pre-sim values of P and X are different from their values in the UDC file.

If you need to follow the prices as levels variables, you may need to read the prices at the beginning of the simulations from the data files and update the value at the end of the simulation.

8.5.3 Are Values on UDC Files Different From Values Shown in the Updated Data?

The values of those Coefficients which appear on the updated data files should be the same as on the corresponding UDC file.

8.6 Coefficient Values (CVL) Files from Data Manipulation TAB Files for AnalyseGE

You may wish to create a **Coefficient Values (CVL) file** in the following two cases.

- If you have a TABLO Input file which just does data manipulation (see section 6.1.4), you may wish to use AnalyseGE to look at the values of various Coefficients and expressions.
- If you do not carry out a simulation with a model containing equations, you may wish to use AnalyseGE to look at the values of various Coefficients and expressions (perhaps because you are having difficulties with simulations with the model).[135]

If so, you can produce a Coefficient Values (CVL) file by inserting the statement

CVL file = <file name> **;** ! no suffix

in your Command file. You can then load this CVL file into AnalyseGE (see section 2.6 of GPD-4). Inside AnalyseGE you can do the same things as when you load a Solution file except that, in the case of a CVL file, there are no Variables or Equations, just Coefficients and Formulas.[136]

CVL files have suffix **.CVL** . You can use <cmf> in your CVL file statement (see section 2.5.4). You can also use the special statement

CVL file = yes ;

which is the same as "CVL file = <cmf> ;" to create a CVL file with the same name as the Command file.

[135] Put the statement "simulation = no ;" into your Command file if you don't want a simulation carried out.

[136] CVL files were introduced in Release 8.0.

Examples of "cvl file = " statements are:

```
cvl file = sjchk ;  ! will produce file sjchk.cvl

cvl file = <cmf> ;  ! will produce file  <cmf>.cvl where <cmf> is replaced by the Command file stem.
                    ! If your Command file is c:\oranig\test3.cmf this will produce c:\oranig\test3.cvl
```

See section 6.8.2 for a detailed example of the use of a CVL file with AnalyseGE. When you use AnalyseGE to look at the values of Coefficients on a CVL file, the values you see are those **at the end of the TAB file**. See section 2.6.1 of GPD-4 for details.

CVL files are similar to SLC files (see section 8.4 above). However, unlike SLC files, CVL files are not exactly Header Array files since they contain the TABLO Input file and Command file in the non-Header-Array format expected by the program TEXTBI (see chapter 14 of GPD-4), which can be used to extract these from a CVL file. However most of the data in a CVL file is in Header Arrays. Accordingly you can use ViewHAR (see section 2.2 of GPD-4) to look at most of the data in a CVL file. If you do so, you will need to select option **Ignore bad headers** under ViewHAR's ***File* | *Options…*** menu item.

8.7 Coefficient Values (CVL) Files For Analysing Why a Simulation Crashes

A simulation may crash because of a singular matrix (see section 15.1) or because of arithmetic problems including division by zero (see sections 15.3 to 15.6). In such cases, you may want to load the values of all Coefficients into AnalyseGE (or ViewHAR) to try to understand what is going wrong.

To do this, you can create a CVL file (see section 8.6 above) from a TAB file containing equations if you instruct GEMSIM or the TABLO-generated program not to do a simulation. That is, put the lines

```
simulation = no ;
CVL  file = … ;
```

into your Command file to create a CVL file containing the values of all coefficients from a model which contains equations.

If you condensed your model when running TABLO, you may wish to see the values of the system-initiated coefficients (the ones introduced by TABLO during condensation – see section 2.3.2 of GPD-2) in the CVL file. If so, you can include the statement

```
CVL system_coefficients = yes ;
```

in your Command file. Then all the values of all system-initiated coefficients (for example, C00134) are available in the CVL file. See section 8.4.2 above for advice about seeing the values of these system-initiated coefficients in AnalyseGE.

If you use AnalyseGE to load a CVL file produced from a TAB file containing variables and equations, there are no values available for the variables (since you did not carry out a simulation). However, AnalyseGE lets you select and evaluate expressions involving Coefficients from Equations in the TABmate form.

CHAPTER 9

9. Equations Files and BCV and LU Files

Equations files (section 9.1) contain numerical versions of the linearized equations of the model. They may be created during a simulation.

Base Coefficient Values (BCV) files (section 9.2) contain values of the Coefficients (using the pre-simulation data base). These files are used by GEMSIM and TABLO-generated programs when doing the updates. They are usually deleted before the end of the simulation but may be saved.

LU files (section 9.3) contain details of the closure and numerical details of the LU decomposition of the LHS Matrix.

9.1 Equations Files

Equations files contain numerical versions of the linearized equations of the model (that is, contain the Equations Matrix as described in section 2.13.1 of GPD-1). The values of the coefficients used in the equations are those in the pre-simulation data. Equations files may be created during a simulation.

To create an Equations file, use the statement

```
Equations file = <file_name> ;   ! Example. Equations file = sj ;
```

The default is not to create an Equations file, so none will be created unless you include that statement.[137]

Usually an Equations file is not required. However you may want to create one

- if you are having problems with the closure (since then the error message may suggest that you run the program SUMEQ – see chapter 13 of GPD-4), or
- if you are having problems with the homogeneity of your model (see section 13.2 of GPD-4).

You will need to create an Equations file if you wish to use SAGEM to carry out Johansen simulations (see section 2.12 of GPD-1 and chapter 10 below).

9.1.1 Using an Equations File in SAGEM

If you wish to run SAGEM, you must first create the Equations file by running GEMSIM or the TABLO-generated program for your model, using the statement

```
equations file = <file_name> ;
```

Then you can include the statement

```
use equations file  <file_name> ;
```

in your Command file when you run SAGEM.

See sections 2.12.1 and 2.12.2 of GPD-1 for an example.

Further details about SAGEM are given in chapter 10.

[137] In Release 5.2 and earlier the default was to save an Equations file. This default was changed in Release 6.0 if a Command file was used (but not otherwise). In Release 7.0, the default is not to save an Equations file, whether or not a Command file is used. This means that it is no longer possible to save an Equations file unless you use a Command file and you include an "equations file =" statement.

9.1.2 Differences in Step 1 Depending on Whether or Not Equations File is Saved

If an Equations file is being saved, it is saved during step 1 of a multi-step calculation. Hence whether or not an Equations file is being saved affects how the program proceeds in step 1 of a multi-step calculation (but does not affect the other steps).

In order to save an Equations file, it is necessary to calculate all the separate entries in the Equations matrix (that is, the matrix C in equation (1) in section 2.13.1 of GPD-1). Once the matrix C has been calculated, when the closure is known, we can divide the matrix C into matrices A and D as in the equation

$$\mathbf{A}\,\mathbf{z}_1 = -\,\mathbf{D}\,\mathbf{z}_2$$

where $\mathbf{z}_1$ and $\mathbf{z}_2$ are respectively the vectors of endogenous and exogenous variables (see equation (2) in section 2.13.2 of GPD-1). When the shocks are known, it is easy to form up the matrix A and the vector b in the equation

$$\mathbf{A}\,\mathbf{z}_1 = \mathbf{b} \qquad (*)$$

(this is equation (3) in section 2.13.2 of GPD-1) which must be solved at each step of a multi-step simulation.

If you do not want to save an Equations file, the program bypasses the full C matrix and heads directly to the equation (*) above on step 1 of a multi-step calculation. This is also what normally happens at steps 2,3... of a multi-step calculation (see section 6.2).

One advantage of heading straight for (*) is that memory is only required to store all the separate entries of A, which are the columns of C corresponding to endogenous variables. Especially if your model has a large number of exogenous variables, the memory for storing matrix A may be significantly less than the memory required to store all of C.

When the program heads straight for (*), the calculations of the different submatrices (see chapter 13 of GPD-4) distinguish between entries of C corresponding to exogenous and endogenous variables. The number of nonzeros on the LHS (that is, in the matrix A) and the number contributing towards the vector b (that is, corresponding to shocked exogenous variables) are reported separately. In these calculations, entries of C corresponding to exogenous variables which are not shocked are ignored.

9.1.3 Model Name, Version and Identifier

The topics covered in this section are less important than when they were introduced. Now models are managed by managing their TABLO Input files rather than by specifying model names, versions and identifiers. This was a feature introduced several years ago to try to help users distinguish between different models. Associated was the version number and version identifier. These are not important in the working practice of most modellers and we have adopted the default that the MODEL name will be taken as blank if no "model = … ;" statement is present. The defaults are for the MODEL name to be all blanks, the version number to be 1 and the version identifier to be all blanks.

The main practical consequence of this is that you don't ever need to include a statement "model = … ;" in your Command files, or give a version number or model identifier if you do not want to.

However if you wish to, when you create an Equations file using the statement "Equations file = …. ;" you may specify a model name (up to 8 characters), a model version number (an integer) and a model identifier (up to 60 characters). These are recorded on the Equations file and are also put on any Solution file produced from this Equations file. They then appear as a description of the model in simulation results (for example, on a GEMPIE Print file).

Model = <name> ; ! If this statement is omitted, model =< blanks> ; is assumed

version = <integer> ; ! default 1

identifier = <identifier> ; ! default is all blanks

The model identifier has only a descriptive role, but the model name and version number are also used for checking, as described below.

If you use a closure stored on an Environment, Solution or LU file (see sections 5.2 and 9.3), the model name and version number on this file are compared with those on the Equations file being used or created. If they are

different a warning is issued, but you are still allowed to use the stored closure provided that the variables in the model, their names and numbers of components agree between the two files.

You may wish to use an LU file – that is, to take both the closure and LU decomposition of the left-hand-side matrix from this file (see section 9.3), perhaps via a "use LU file … ;" statement in a Command file. This will be allowed if the model name and version number on the LU file agree with those on the Equations file being used.[138] If they do not agree, you will only be permitted to take the closure from the LU file and the LU decomposition will be calculated from scratch.

9.2 Base Coefficient Values (BCV) Files

Base Coefficient Values (BCV) files contain values of the Coefficients (using the pre-simulation data base). These files are used by GEMSIM and TABLO-generated programs when doing the updates. They are usually deleted before the end of the simulation but may be saved.

We suggest that you never keep the BCV file since they can take up a lot of disk space and they are of no real use.

To create a BCV file, use the statement

```
bcv file = <file_name>   ;   ! Example. Bcv file = sj ;
```

The default is not to create a BCV file, so none will be created unless you include that statement.[139]

9.2.1 Starting from Existing Equations and BCV Files Not Recommended

In the past, some modellers saved an Equations file and the associated BCV file from a simulation and started from these to attempt to speed up other simulations which start from the same pre-simulation data (see section 12.4.2). Although the use of these can speed up simulations a little, supporting this feature has become very costly to us (the developers) and has greatly increased the risk of bugs. Accordingly we are seriously considering not supporting this in the next release of GEMPACK.

Also, starting from existing Equations and BCV files disables some desirable features. For example,

- levels results are not available (see section 8.3).
- "extra" statements are not allowed (see section 6.6).
- user-specified accuracy may not be allowed (see the note in the second paragraph of section 7.4.2).
- an SLC file (see section 8.4) is not created so you cannot use AnalyseGE (see section 2.6 of GPD-4).

If you do start simulations from existing Equations and BCV files, you should read section 12.4.2 carefully.

---

[138] You can only use an LU file (in the above sense) when you are using an existing Equations file. If you are creating a new Equations file via GEMSIM or a TABLO-generated program, you can take the closure from an old LU file but will not be allowed to read the LU decomposition itself from such a file.

[139] In Release 5.2 and earlier the default was to save a BCV file. This default was changed in Release 6.0 if a Command file was used (but not otherwise). In Release 7.0, the default is not to save a BCV file, whether or not a Command file is used. This means that it is no longer possible to save a BCV file unless you use a Command file and you include an "BCV file =" statement.

9.3 LU Files

LU files contain details of the closure and numerical details of the LU decomposition (see section 12.1) of the LHS Matrix.

These files were introduced as a means of speeding up simulations when users wanted to carry out several simulations (via GEMSIM, a TABLO-generated program or SAGEM) with the same base data and closure (but different shocks). Section 12.4.2 explains how the use of an LU file could speed up step 1 of a multi-step calculation with GEMSIM or a TABLO-generated program.

We no longer recommend the use of LU files with GEMSIM or TABLO-generated programs for the reasons set out in section 9.2.1 above. [In order to use an LU file with GEMSIM or a TABLO-generated program, you must also be using existing Equations and BCV files.] We use a smaller font for the documentation of LU files below to reinforce the idea that, in our opinion, you should not be using these files.

- You can save an LU file when running GEMSIM, a TABLO-generated program or SAGEM via the command

  save LU file <file_name> ;

- You can use an LU file when running GEMSIM, a TABLO-generated program or SAGEM via the command

  use LU file <file_name> ;

  When you use an LU file, the closure and the LU decomposition are read from that file.

- Since an LU file contains details of the closure, you can take the closure from it or modify the closure on it via the statements

  take closure from LU file <file_name> ;

  modify closure from LU file <file_name> ;

  In either of these cases, only the closure is taken from the LU file. The LU decomposition is calculated afresh. So these uses of an LU file essentially the same as the corresponding uses of an Environment file as documented in sections 5.2.2 and 5.2.3 above.

For other information about LU files, see section 9.1.3 above and section 12.4.2 below.

CHAPTER 10

10. Several Simultaneous Johansen Simulations via SAGEM

In this chapter we go into more detail about the contents of Solution files, particularly those produced by SAGEM, and introduce the idea of individual column solutions and subtotal solutions. We also describe in a little more detail how the linearized equations are solved. In the last section we explain why individual column solutions cannot be obtained in a multi-step simulation carried out using GEMSIM or a TABLO-generated program.

SAGEM must use an Equations file – see section 9.1.1.

10.1 The Solution Matrix

As explained in section 2.12 of GPD-1, it is possible to calculate the results of several Johansen simulations in one run of SAGEM. In this section we explain this in a little more detail.

As you have seen in section 2.13.2 of GPD-1, once the exogenous/endogenous split has been chosen, the system of linearized equations of a model with n equations and m variables

$$\mathbf{C\ z = 0} \qquad (1)$$

becomes

$$\mathbf{A\ z_1 = -\ D\ z_2} \qquad (2)$$

where z_1 and z_2 are respectively the vectors of endogenous and exogenous variables in the system,

A is n x n and D is n x (m – n).

A is called the **LHS Matrix** (Left Hand Side Matrix). The columns of A and D are those columns of C corresponding respectively to endogenous and exogenous variables. If k of the m – n exogenous variables are given nonzero shocks, the problem of finding the change in each of the n endogenous variables resulting from each of these k shocks reduces to solving, for X, the matrix equation

$$\mathbf{A\ X = B} \qquad (3)$$

where B is an n x k matrix, the jth column of which is obtained by multiplying the column of D corresponding to the jth shocked variable by the negative of the shock given to this variable.

When the closure is economically sensible, the matrix A is invertible and the equation (3) can be solved.

The n x k matrix X is the "solution" of the simulation: its entry

`x(i,j) (in row i and column j)`

shows the percentage change (or change) in the ith endogenous variable resulting solely from the shock given to the jth shocked variable. Because we are dealing with a system of linear equations in (3), the sum of all entries in any row I of X can be taken as the percentage change in the ith endogenous variable resulting from **all** of the k shocks given. The elasticity of the ith endogenous variable with respect to the jth shocked variable can be obtained by dividing x(i,j) by the shock given to the jth shocked variable. (The first two columns of results in Table 2.12.3 in section 2.12.3 of GPD-1 are the X matrix, of size 27 x 2, for the simulation carried out there.)

The n x k matrix X is sometimes called the **Solution Matrix** of the Johansen simulation.

The Solution Matrix X and Totals Column T

	Shocked variables				
	1	2	…	k	Totals column
Endogenous variables					
1	[x(1,1)	x(1,2)	…	x(1,k)]	[t(1)]
2	[x(2,1)	x(2,2)	…	x(2,k)]	[t(2)]
:	:	:	…	:	:
:	:	:	…	:	:
n	[x(n,1)	x(n,2)	…	x(n,k)]	[t(n)]
	Solution Matrix X				**Totals column T**

10.1.1 Individual Column Results

When SAGEM is used in this way, the Solution file produced contains selected portions of the Solution Matrix X. The exogenous variables corresponding to the columns of X retained on the Solution file are called the **individually-retained exogenous** variables. (There may be none of them.) The endogenous variables corresponding to the rows of X retained in the Solution file are called the **individually-retained endogenous** variables. If any of the individual columns of the matrix X are retained on the Solution file (that is, if there are any individually-retained exogenous variables), we say that the Solution file contains **individual column results**.

The Command file statements for retaining individual column results are:

```
individually-retained exogenous <list> ;
individually-retained endogenous <list> ;
```

Use the procedure in section 5.4.1 to specify <list>. If you do not include an "individually-retained exogenous" statement, no individual columns will be put on the Solution file. If you have any individual columns, but do not include an "individually-retained endogenous" statement, all endogenous variables are individually-retained.[140]

See section 5.4.2 for an example of the use of these statements.

10.1.2 Cumulative or Row Totals Results

It is also possible for the Solution file to contain all or some of the rows of the n x 1 vector T whose ith entry (for I between 1 and n) is obtained by adding up all the entries in row I of the matrix X. The endogenous variables corresponding to the rows of T (if any) retained on the Solution file are called the **cumulatively-retained endogenous variables** since the ith entry of T is the cumulative effect on the ith endogenous variable of all the shocks applied. If some of T is retained on the Solution file, we say that the Solution file contains **cumulative** (or **row totals**) **results**. (The third column of results in Table 2.12.3 in section 2.12.3 of GPD-1 is the T vector for the simulation carried out there.)

The related Command file statement is:

```
cumulatively-retained endogenous  <list> ;
```

Use the procedure in section 5.4.1 to specify <list>. If you do not include an "cumulatively-retained endogenous" statement, all endogenous variables are cumulatively-retained.[141]

See section 5.4.2 for an example of the use of these statements.

---

[140] That is, the defaults are:
none for the set of individually-retained exogenous variables, and
all endogenous variables for the set of individually-retained endogenous variables.

[141] That is, the default for the set of cumulatively-retained endogenous variables is all endogenous variables.

10.1.3 Subtotal Solutions

When you run SAGEM, if you choose to retain some of the rows of the totals vector T, you can also store on the Solution file selected **subtotals** which are the cumulative effect of several (but not necessarily all) of the shocks. We call these **subtotal solutions**. Each of these can be thought of as an n x 1 vector whose ith value is obtained by adding all entries in row I of X which are in columns corresponding to the shocks in question. The rows kept on the Solution file for each subtotal are the same rows as those kept from T, namely those corresponding to the cumulatively-retained endogenous variables. More details about obtaining subtotals results (which can also be obtained when running GEMPIE) are given in section 10.2 below.

The related Command file statement is:

```
subtotal <list> = <description> ;
```

Use the procedure in section 5.4.1 to specify <list>. <description> can be any character string up to 77 characters long. See section 10.2 for examples.

10.1.4 Contents of a SAGEM Solution File

A Solution file produced by SAGEM may contain both individual column and totals/subtotals results, or just one of these two types, according to the instructions you issue while running SAGEM.

In summary, the Solution file produced by SAGEM contains

- selected rows and columns of the Solution Matrix X, showing the effects on selected endogenous variables of selected shocks, and/or
- selected row totals (or subtotals) across all (or some) columns of X, showing the total effect, accumulated over all (or some) of the shocks, on selected endogenous variables.

You control the contents of the Solution file (when running SAGEM) by specifying

a) whether the Solution file contains individual results, totals/subtotals results or both types of results,

b) the individually-retained exogenous variables,

c) the individually-retained endogenous variables,

d) the cumulatively-retained endogenous variables,

e) which subtotals (if any) and the effects of which shocks are included in each subtotal solution.

We strongly recommend that you always use a Command file when running SAGEM. [Running SAGEM interactively is error-prone and difficult to document or reproduce.]

10.1.5 Viewing the Results of a Simulation

When you run GEMPIE you can translate to printable form any results stored on the Solution file you access. If the Solution file contains both individual column results and totals/subtotals results, you can only convert part of one of these in any one run of GEMPIE. If you want to print both sorts of results, you must run GEMPIE twice. (See chapter 7 of GPD-4 for more information about GEMPIE.)

Note that ViewSOL (see section 2.3 of GPD-4) cannot be used to access individual column results.

If the Solution file contains individual column results, when you run GEMPIE, you can use these individual column results to form up subtotals results (the results of all or some of the shocks whose individual effects are stored) and then convert these subtotals results to printable form. More details of this are given in section 10.2 below.

10.2 Subtotals

A subtotals result is one showing the effect on the endogenous variables of a single shock or a group of shocks. An example will make this clear.

Suppose you shock

- the foreign currency prices [p(i),i=1,…,10] for the imports of each of the 10 commodities by 5 per cent,
- the real wage rate [w] by 2 per cent, and
- household consumption [c] by 1 per cent.

Then the Solution Matrix X (see section 10.1 above) would have 12 columns (10 for the p(i) shocks and one each for the w and c shocks). The combined effect of just the p(i) shocks would be given by adding just the first 10 columns of X – this is one subtotal you may wish to calculate and print. A second subtotal solution of the model would be the combined effects of the w and c shocks.

Within GEMPACK it is possible to calculate subtotals results in the context of a Johansen simulation (carried out via SAGEM) or in the context of a multi-step simulation carried out via GEMSIM or a TABLO-generated program.[142] In this section we discuss subtotals in the context of a Johansen simulation. Subtotals in the context of a multi-step simulation are described in chapter 11.

Within the context of a Johansen simulation, there are two ways of forming subtotals results; the first is when running SAGEM and the second is when running GEMPIE. These alternatives are described in sections 10.2.1 and 10.2.2 below. Then, in sections 10.2.3 to 10.2.5, we give more general information relevant to subtotals.

10.2.1 First Method using SAGEM

When you run SAGEM to carry out the simulation in the example above, you can ask it to store on the Solution file all of the following:

a) the 12 individual columns,
```
individually-retained exogenous %all ;
```

b) the cumulative total T,
```
cumulatively-retained endogenous %all ;  ! this is the default
```

c) the subtotal solution showing the combined effect of the 10 p(i) shocks,
```
subtotal p = foreign currency price changes ;
```

d) the subtotal solution showing the combined effect of the w and c shocks
```
subtotal w  c = wage and household consumption changes ;
```

Then, when you run GEMPIE, you will be able (in one run) **either** to

- print some or all of the individual columns, **or**
- print the cumulative solution column T (see section 10.1.2 above) and 0, 1 or 2 of the subtotal solutions on the Solution file.

You can see the cumulative solution T and any of the subtotals solutions when you run ViewSOL. However, ViewSOL cannot show individual column results.

[142] Subtotals in the context of a Johansen simulation have been a part of GEMPACK since its early days. Subtotals in the context of a multi-step simulation were introduced in Release 7.0.

Another example (which you can carry out using the example files supplied with GEMPACK) is shown in section 5.4.2 where three subtotals results are set up in SAGEM Command file MOSAGEM.CMF. You might like to run this simulation via SAGEM[143] and then look at the subtotals results via GEMPIE or ViewSOL (see section 10.2.5 below).

When you use SAGEM to calculate subtotals results, the subtotals solutions are permanently stored on the Solution file, where they can be easily viewed via GEMPIE or ViewSOL whenever required, simply by referring to their subtotal number. This method is suitable when you have a number of subtotals that you want to refer to often.

10.2.2 Second Method using GEMPIE

Alternatively, you may run SAGEM to carry out the example above (shocks to foreign currency prices, the real wage and household consumption), but just store the 12 individual columns (but no cumulative total or subtotals) on the Solution file. The Command file statements to achieve this are

```
Individually-retained exogenous %all ;
Cumulatively-retained endogenous %none ;
```

You can still obtain the same two subtotal solutions as in section 10.2.1 above by running GEMPIE and specifying two subtotals to be printed, the first summing over the 10 foreign currency price shock columns and the second over the w and c shocks.

Sections 7.2.2 and 7.2.3 of GPD-4 contain details about using GEMPIE to set up subtotals. Note that Command files cannot be used with GEMPIE, so you set up the subtotals by running GEMPIE interactively by responding to prompts (or using a Stored-input file). Section 7.2.3 of GPD-4 contains a detailed example which you can carry out using the GEMPACK example files.

GEMPIE can only calculate subtotals results from any individual column results stored (by SAGEM) on the Solution file. If there are none, GEMPIE cannot calculate any subtotals results.

When you use GEMPIE to calculate subtotals results, the subtotals are not stored on the Solution file (since GEMPIE never alters a Solution file). This method is suitable when you do not know in advance what subtotals you may require from a particular simulation.

10.2.3 Subtotals and Sets of Shocks in General

For a given simulation, you can have several different subtotals. Each is associated with a subset of the shocks. To specify a subtotal, you must say which of the shocked variables you want the subtotal to reflect. (This specifies which columns of the Solution Matrix X in section 10.1 above are to be added up to produce the subtotals column.)

10.2.4 Subtotal Descriptions

Each subtotal has an associated **subtotal description** which you supply to remind you and others what the subtotal shows. These descriptions can be up to 77 characters long.

When you set up a subtotal in SAGEM (see section 10.2.1), the description is given after the "=" in the "subtotal" statement.

When you set up a subtotal using GEMPIE (see section 10.2.2), you enter the description in response to a prompt.

The subtotal description is shown when you view the subtotal results via GEMPIE or ViewSOL.

---

[143] The Command file MOSAGEM.CMF requires an Equations file and an Environment file. These are both created if you run GEMSIM or the TABLO-generated program for Miniature ORANI (MO.TAB) to carry out the simulation in Command file MOTAR.CMF which is supplied with the GEMPACK examples. So you should run MOTAR.CMF before trying to run SAGEM via MOSAGEM.CMF.

10.2.5 Viewing Subtotals or Individual Column Results using GEMPIE or ViewSOL

Any individual results, totals results or subtotals results on the Solution file can be printed when running GEMPIE. You can also set up and print new subtotals results from the individual columns results stored (by SAGEM) on the Solution file, as described in section 10.2.2 above.

You can use ViewSOL to access totals and subtotals results produced using SAGEM, but you cannot access individual column results via ViewSOL.

A hands-on example illustrating these points can be found in section 2.5.6 of GPD-8.

10.3 No Individual Column Results from Multi-Step Simulations

Although it is possible to carry out several Johansen simulations (all with the same closure) simultaneously in one run of SAGEM, it is not possible to carry out several multi-step simulations in one run of GEMSIM or the appropriate TABLO-generated program, as explained below.

For Johansen simulations, the matrix A in (2) (near the start of this chapter) depends only on the closure and is not affected by the shocks. Since the main computing cost of a Johansen simulation is that of calculating the LU decomposition of A (see section 12.1), it is possible to carry out several different simulations (all with the same closure but different sets of shocks) in one run of SAGEM. Certainly this increases the number of right-hand-sides to solve for, but the solution for each of these is relatively cheap to compute.

In contrast, when you give a set of shocks in say a 2-step simulation (see Figure 2.13.3 in section 2.13.3 of GPD-1), the program

- first applies half of each shock,
- then updates the data to reflect the effects of these,
- then (for the second of these steps) recomputes the matrix C (as in equation (1) in section 10.1 above) and
- solves for the effects of the next half of the shocks in question.

For 2 different sets of shocks (that is, for two different multi-step simulations), you would get two different C matrices after the first half of each shock has been applied. (In the notation of Figure 2.13.3, if you have two sets of shocks you will reach two different points C2 after the first step and then move in two different directions from these in the second step.) This means that there are two different A matrices to LU decompose, and it is not feasible to carry on these two quite separate calculations side-by-side.[144]

Of course, this problem would become even less tractable for larger numbers of steps.

For this reason, the Solution file from a multi-step simulation contains just one totals solution (the cumulative effect of all the shocks). No individual column results can be present. However, perhaps surprisingly in the light of the above, the Solution file can contain subtotals results (see chapter 11).

[144] Two different sets of shocks with the same closure give rise to the same A matrices in the first step of a multi-step simulation. When computing an extrapolated solution from, say 1-step and 2-step solutions, GEMSIM or TABLO-generated code takes advantage of this by computing the solution of the first step of the 2-step computation immediately after computing the solution of the first step of the 1-step computation, both using the same LU decomposition.

CHAPTER 11

11. Subtotals via GEMSIM or TABLO-generated Programs

A subtotals result is one showing the effect on the endogenous variables of a single shock or a group of shocks.

This chapter describes what subtotals results mean and how they can be calculated within the context of a multi-step simulation carried out using GEMSIM or a TABLO-generated program. Subtotals can also be calculated within the context of a Johansen simulation (see section 10.2).

11.1 Subtotals from GEMSIM and TABLO-generated Programs

It is possible to decompose simulation results from a multi-step solution with respect to exogenous shocks.[145] The theory and motivation underlying this are discussed in detail in the paper by Harrison, Horridge and Pearson (2000). We refer to this paper as **HHP** in this chapter.

The cumulative solution determines the path of the solution and coefficients are updated from the data in the cumulative solution at each step. However, the separate contributions of the exogenous shocks are recorded and accumulated to find the effect of just one particular shock or a combination of several shocks in a subtotal. This is similar to a subtotal in SAGEM as discussed in chapter 10. The difference is that whereas each subtotal was a solution of the Johansen simulation for SAGEM, when you are using a multistep solution in GEMSIM or TABLO-generated program, there is only one simulation solution, the cumulative solution. The subtotal solutions are just the effect of the subtotal shocks **within the context of the cumulative solution**. This is a slightly different meaning to that for subtotals produced by SAGEM; however the same syntax is used in both cases.

Subtotal statements in Command files produce subtotal solutions on the Solution file. The Command file syntax is the same as for subtotals produced by SAGEM – see section 10.1.3. See sections 11.3 and 11.4 below for examples.

11.2 Meaning of Subtotals Results

Consider an application in which there are several shocks to tariffs and also several shocks to technology variables (say, technological improvement in various parts of the model). It would be natural to calculate two subtotals results, the first the consequence of just the tariff changes and the second the consequence of the technology changes. In this case, for each endogenous variable, the results of these two subtotals would add exactly to the overall simulation result for that endogenous variable (as shown in the cumulative solution). For example, if the subtotals results for real consumption were 3.0 for the tariff changes and 2.5 for the technology changes, the overall simulation result for real consumption would be exactly 5.5 (the sum of these two). This applies to percentage change results or to change results. It would be natural to attribute 3.0 per cent change in real consumption to the tariff shocks and 2.5 per cent to the technology shocks.

The above is true for several groups of shocks. This is the main property of the decomposition (as described in HHP) which states that:

> **If the set of exogenous shocks is partitioned into several mutually exclusive and exhaustive subsets then, for each endogenous variable Z, the change or percentage change in Z is exactly equal to the sum of the contributions of these subsets of exogenous shocks to the change or percentage change in Z.**

What is called in this statement "the contribution of a subset of shocks to the change or percentage change" is what we are calling in this chapter the **subtotals result corresponding to this group of shocks**.

---

[145] This was introduced in Release 7.0.

In the simple example in section 11.3 below, there are just 2 shocks and two subtotals results (the consequences of each shock separately). For any endogenous variable, the sum of its values in these two subtotals results will exactly equal its value in the cumulative solution.

Note that it makes perfect sense to calculate and report subtotals results which do not add to the cumulative result. For example, in the tariff and technology example above, it would also make sense to calculate extra subtotals being the consequences of the tariff shocks made by one or more of the regions in the model. It is natural to attribute these subtotals results to the tariff reform in just the relevant region.

11.3 Subtotal Example

As a simple example of subtotals, you can add two subtotals to the basic Stylized Johansen simulation discussed in section 2.4 and 2.6 of GPD-1. Below we ask you to change the shocks to give a 10 percent shock to labor and a 20 percent shock to capital supply. The first subtotal is just the effect of the labor supply shock, the second subtotal is the effect of the capital supply shock.

Copy the Command file SJLB.CMF (see section 2.8.1 of GPD-1) to a new file called SJSUB.CMF and edit it to make the changes shown in Bold below.

```
! The following GEMPACK Command file called SJSUB.CMF
!  carries out a multi-step simulation
!  for the Stylized Johansen model.

!  Auxiliary files (usually tells which TAB file)
auxiliary files = sj ;

! Data files
file iodata = sj.dat ;
updated file iodata = <cmf>.upd ;

! Closure
exogenous p_xfac ;
rest endogenous ;

! Solution method information
method = euler ;
steps = 1 2 4 ;

! Simulation part

! Name of Solution file is inferred from name of Command file.
! (See section 2.5 of GPD-3.)

shock p_xfac  = 10  20 ;

verbal description =
Stylized Johansen model. Standard data and closure.
10 per cent increase in labor
20 percent increase in capital;

! Subtotals
subtotal p_xfac("labor") = Effect of labor supply change ;
subtotal p_xfac("capital") = Effect of capital supply change ;
```

Command file SJSUB.CMF

Run the new Command file in the usual way described in section 2.4 or 2.6 in GPD-1.

You should find, for example, that the simulation result for p_Y (household consumption) is an increase of 13.8958 per cent. The labor and capital subtotals results are 6.1105 and 7.7853 respectively. Note that these add to 13.8958 as expected.

11.4 More Substantial Examples

HHP describes in detail how subtotals results can be used to decompose the results of a well-known trade reform application with GTAP. We encourage GEMPACK users interested in calculating subtotals to decompose results for their own model to read and understand the application in HHP first. This example is an ideal introduction to subtotals and decomposing results in the context of a multi-step simulation. You can carry out this application for yourself using the example model files (including Command file GIP73A.CMF) supplied with GEMPACK, as described in detail in section 2.6.11of GPD-8.

Another GTAP example showing subtotals can be seen in the GEX15.CMF simulation in section 7.1.1. Here there are 3 subtotals which decompose this liberalization into separate output, import and export liberalization effects.

A decomposition of multi-step results in the context of an application with ORANI-G is described in section 2.8 of GPD-8. Again you can carry out this application by working through that section in GPD-8, using Command file ORNGAPP1.CMF (see section 2.8.7 of GPD-8).

11.5 Processing Subtotal Results

GEMPIE can process and show these subtotals results. In any one run of GEMPIE, you can print either the cumulative solution plus any subtotals, or the cumulative solution plus any levels results. However you cannot print both subtotals and levels results in the same run of GEMPIE. See section 7.2.1 of GPD-4 for more details. If you run GEMPIE via WinGEM, WinGEM will ask you which solutions you wish to print if relevant.

The current versions of the Windows programs ViewSOL, RunGEM and RunGTAP can all show subtotals results. Users of RunGEM and RunGTAP (see section 2.5 of GPD-4) can use point and click methods to select the shocks in each subtotal result.

SLTOHT (see chapters 8 and 9 of GPD-4) can access subtotals results. SLTOHT offers access to subtotals (in addition to the cumulative solution) in appropriate circumstances – see, for example, section 8.2.4 of GPD-4. See also SLTOHT's option **SEP**, which is documented in section 8.9 of GPD-4.

The programs RWSOL and MKSOL (see section 11.2 of GPD-4) are able to process subtotals solutions.

Subtotal results can be accumulated using the program ACCUM (see section 10.2.4 in GPD-4) and differences can be calculated using DEVIA (see section 10.3.3 in GPD-4).

11.6 Subtotals Results Depend on the Path

When GEMSIM or a TABLO-generated program calculates subtotals results, it is doing so following a path through exogenous space in which the rate of change in each exogenous variable is the same everywhere on the path. [As explained in section 3.5 of HHP, this corresponds to a straight line in n-dimensional space (if there are n shocks) from the pre-simulation levels values of the exogenous variables to their post-simulation levels values.]

If alternative paths are followed, the subtotals results may be different.

This can be seen easily, as the examples in section 3.4 of HPP show. We encourage you to read this part of HHP in order to see why subtotals results depend on the path.

The example in HHP is a mathematical system of equations rather than an economic model.

Accordingly we give in section 11.6.1 below an example based on the SJSUB.CMF simulation in section 11.3 above. The details of this example are somewhat complicated. You do not need to be understood the example below in order to use subtotals in your modelling work. Accordingly **you may prefer to skip section 11.6.1**.

More details about this and other aspects of contributions and decompositions can be found in HHP.
HHP contains results for alternative decompositions for a well-known GTAP application.

11.6.1 Example Showing How the Path Can Affect Subtotals Results

The subtotals results for p_Y (percentage change in income) from the SJSUB.CMF simulation in section 11.3 above are 6.1105 (effect of the labor supply change) and 7.7853 (effect of the capital supply change). These come from the **straight-line path** in 2-dimensional exogenous space in which these supplies are both increasing at a constant rate.

Now consider a **second path** through exogenous space in which first supplies of labor are increased by 10 percent holding capital fixed, and then supplies of capital are increased holding labor supply fixed (at 10 percent above its pre-simulation value). This ends at the same point in exogenous space as the straight-line path in the paragraph above. However the p_Y subtotals results for this second path would be different, as explained below.

To work out these subtotals results, consider firstly the results of two related simulations.

- In the first simulation [SIM1], labor supply is increased by 10 percent, holding capital fixed. [This simulation moves in exogenous space along the first part of the second path above.] The p_Y result for this simulation is 5.8853 (the same as the p_Y result from the SJLB.CMF simulation).
- The second simulation [SIM2] starts from the updated data after this first simulation and increases the supply of capital by 20 percent, holding labor supply fixed. [This simulation moves in exogenous space along the second part of the second path above.] If you solved this simulation, you would find that the p_Y result is 7.565.

Note that the p_Y results from these two simulations – 5.8853 and 7.565 – are not a decomposition of the p_Y result for the SJSUB.CMF simulation since 5.8853 and 7.565 do not add to the cumulative p_Y result of 13.8958 from SJSUB.CMF. However, these two percentage changes do compound to 13.8958 as you should expect.[146]

Hence the p_Y subtotals results from the second path above (the one where all of the labor supply change happens first, then all of the capital supply change), must be

5.8853 (effect of the labor supply change),

8.0105 (effect of the capital supply change).

The labor supply change subtotal must be equal to the p_Y result (5.8853) of the first simulation [SIM1] above. The capital supply subtotal result must therefore be 8.0105 since (from the theory in HPP) we know that the two subtotals results must add to the cumulative p_Y result from SJSUB.CMF of 13.8958.[147]

Note that these two subtotals results are different from those from the SJSUB.CMF simulation, namely

6.1105 (effect of the labor supply change),

7.7853 (effect of the capital supply change).

when the path through exogenous space is the usual straight line.

You would get different subtotals results again if you considered a **third path** through exogenous space in which first capital supply was increased by 20 percent (holding labor supply fixed) and then labor supply was increased by 10 percent (holding capital supply 20 percent above its pre-simulation value).

These three paths are similar to paths in the example in section 3.4 of HPP.

---

[146] Note that 1.058853*1.07565 – 1 = .138955. This corresponds to a percentage change of 13.8955 which is acceptably close to the cumulative 13.8958 p_Y result from SJSUB.CMF. [You would see complete agreement if you solved SJSUB.CMF and the two simulations SIM1 and SIM2 with Gragg's method and 2,4,6 steps. Then the p_Y result from SJSUB.CMF would be 13.8959, the p_Y results of the two simulations would be 5.88529 [SIM1] and 7.56538 [SIM2] respectively. These two compound to 13.8959 as expected.]

[147] The 8.0105 capital subtotal result for p_Y could easily be calculated using program ACCUM. To see this, put the results (including the two subtotals results) from SIM1 and SIM2 into CSV files using SLTOHT with option SSS (see section 9.2 of GPD-4). Then run ACCUM with options SUB and ACC (see section 10.2.4 of GPD-4). You will find that 8.0105 is reported as the accumulated capital subtotal result for p_Y. This is obtained from the formula in section 10.2.4 of GPD-4. The result 8.0105 = 0 + [1 + 5.8853/100]*7.565 where 0 is the capital subtotal result for p_Y from SIM1, 5.8853 is the cumulative p_Y result from SIM1 and 7.565 is the capital subtotal result for p_Y from SIM2.

CHAPTER 12

12. Solving Models and Simulation Time

You can speed up your simulations in various ways: The main ones are:

1. Condense your model when running TABLO as described in section 3.8 of GPD-1.
2. Use a TABLO-generated program instead of GEMSIM for large models. [Some CPU times are reported in chapter 4 of GPD-8.]
3. Try changing from MA48 to MA28, the alternative Harwell subroutines or vice versa. See section 12.1.1 below.
4. Use iterative refinement and ignore the Harwell parameter – see section 12.1.
5. Use a multistep method with 3 multistep solutions and extrapolation (possibly with several subintervals and/or automatic accuracy– see sections 7.3 and 7.4) instead of a large number of steps in a single multistep solution. See sections 7.1 and 7.1.1 for advice about this.
6. If you are just testing a new model, consider using a small aggregated model with few sectors before moving on to full simulations with a highly disaggregated model with many sectors.
7. Do not try automatic accuracy (see section 7.4) initially when you are just testing a new model. Wait until you have checked your simulation results for a few simple simulations. You can improve the accuracy later.

An introduction to the procedure by which multi-step solutions are calculated is given in section 2.13.3 of GPD-1 (with which you should be familiar before reading this chapter). Further information about solution methods is given in chapter 7 above, which includes details of Command files statements for various alternatives. In this chapter we discuss various other aspects of multi-step calculations.

We begin by saying (in section 12.1) more about how GEMPACK solves the linearized equations in any step of a multi-step calculation. Then we give (in section 12.2) an intuitive idea of Gragg' method, and the midpoint method. Section 12.4 discusses how long multi-step calculations may take. Section 12.6 contains a discussion about possible convergence problems you may encounter. Section 12.7 contains some information about the possibly large, temporary work files required by GEMSIM and TABLO-generated programs.

12.1 How GEMPACK Programs Solve Linear Equations

Whether you are doing a multi-step simulation via GEMSIM or a TABLO-generated program, or a Johansen simulation via SAGEM, a critical part of the calculation is to solve a system AX=B of linear equations, where A is the n x n LHS matrix in equation (3) in section 2.13.2 of GPD-1 (see also section 10.1 above).

In the systems of linear equations AX=B (see (3) in section 10.1) that arise from general equilibrium models, the matrix A is a **sparse matrix** in the sense that only a small proportion of its entries are nonzero. GEMPACK solves this matrix equation using the Harwell Laboratories sparse matrix routines (see section 12.1.1 below). These do not explicitly invert the matrix A but calculate the so-called **LU decomposition** (see, for example, section 8.1 of Atkinson (1989) or Stewart (1973)) of A. This is an efficient procedure which allows very large models to be handled directly by GEMPACK.

One feature of the Harwell sparse matrix routines is that there is a trade-off between maintaining sparsity during the calculation and minimising the effects of rounding errors, which are inevitable when doing a large computation on a computer. The Harwell routines ask you to supply the value of a parameter U, referred to as

the **Harwell parameter**, which must be between zero and one. The precise consequences of choosing different values for this parameter are difficult to predict.[148]

Fortunately, it is not of great importance since SAGEM, GEMSIM and TABLO-generated code use a well-known **iterative refinement** technique called residual correction of solutions (see, for example, section 8.5 of Atkinson (1989)) when solving AX=B. Our experience is that this compensates very effectively for any loss of accuracy that may have resulted in its absence from using small values of U.[149]

Accordingly we are confident in recommending that you ignore the Harwell parameter U completely and just use the default value of 0.1.

The value of this Harwell parameter U can be set via the Command file statement

```
Harwell parameter = <value> ;      ! Default value is 0.1
```

The default value (which you get if, as we recommend, you omit this statement) is 0.1.

Iterative refinement usually results in more accurate numerical results. However it does take extra time (though usually only a fraction of that for the LU decomposition). If you feel that the time taken for iterative refinement is excessive, you can suppress it by including the statement[150]

```
NIR = yes ;  ! No Iterative Refinement. [Default is "NO" which means "do iterative refinement".]
```

in your Command file.

12.1.1 Sparse-Linear-Equation Solving Routines MA48 and MA28

GEMPACK uses Harwell Laboratory's subroutines for solving Sparse Linear Equations. There are two alternative versions of these routines the original **MA28** routines and the newer **MA48** routines.

Until Release 5.1, GEMPACK used the Harwell Laboratory's sparse-linear-equation-solving routines MA28. These routines are still available but the default (since Release 5.2) is to use the newer MA48 routines (see Duff and Reid (1993)), which are faster in many cases.

Although the MA28 routines are still available, they will only be used if you request them explicitly. In SAGEM, GEMSIM and TABLO-generated programs, there is an option **M28** which you can select interactively to say that you want to use MA28 routines. In Command files, the statement

```
m28 = yes ;  ! Default is "NO" which means "use MA48".
```

achieves the same result.

Apart from the above, the use of MA48 instead of MA28 requires no further input from modellers, who will probably hardly be aware of the change from MA28 to MA48.

12.1.2 Which to Use? MA48 or MA28

In our experience, MA48 is usually significantly quicker for multi-regional and intertemporal models but is often no quicker than MA28 for single-country models (and may even be slightly slower for these). You can easily experiment with your models to see which is faster.

---

[148] Choosing U close to one minimises the rounding errors, so maximises the accuracy of the solution. However this also increases the time required to produce the solution. Choosing U small (say equal to 0.1) decreases the time taken to calculate the solution but may slightly increase the rounding errors (and hence slightly reduce the accuracy of the results).

[149] In one spectacular instance, we tested a very large model whose homogeneity seemed uncertain judging by results produced with U=0.1 without using residual correction. When U was increased to 0.5, the results were as desired but the running time increased from hours to days. When we tested residual correction with this model, we found that, after residual correction, the U=0.1 solution was at least as accurate as the U=0.5 result; further, the residual correction added only a few seconds to the computation time.

[150] Alternatively, select option NIR (see section 14.1).

12.2 Gragg's Method and the Midpoint Method

Gragg's method and the midpoint method are the same except that Gragg's method does one extra pass at the end, as explained later.

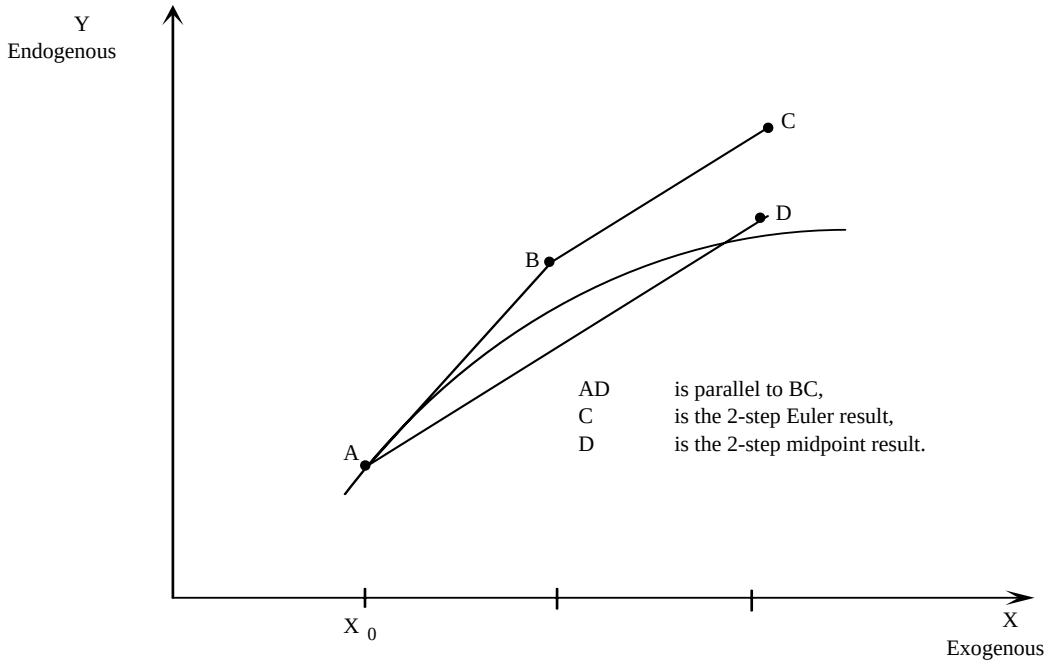
The graphical motivation for Euler's method is shown in Figure 2.13.3 in section 2.13.3 of GPD-1.

At step 1, Gragg and midpoint are identical to Euler's method, following a tangent along the curve from the initial solution. At other steps, all three methods take their direction from the tangent at the current point. The difference is that Euler's method follows this tangent from the current point while Gragg's method and the midpoint method follow this direction but

start from the previous point.

The difference can be seen at step 2, as illustrated in Figure12.2. Each method gets to point B after step 1. At step 2, Euler's method follows line BC (parallel to the curve at point B) and reaches point C after step 2, while Gragg and midpoint follow line AD (also parallel to the tangent to the curve at point B) to reach point D after step 2. You can see that D is significantly closer to the curve than B, which is why, usually, Gragg and midpoint produce more accurate results than Euler does (for the same number of steps). You can experiment with other curves to convince yourself that this is generally the case.

Figure 12.2: Midpoint Method Compared to Euler's Method



Gragg's method and the midpoint method are the same except that Gragg does one more pass. If you select N steps, midpoint does N passes while Gragg does N+1 passes. In the final (N+1)st pass, the Gragg calculation actually starts near the final point and takes the exogenous variable past its end point. This is done to obtain a correction to the result after N passes; the corrected result is usually more accurate than the one after N passes. More details about the theory behind these methods can be found in Pearson (1991) and the references therein. (Gragg's method is sometimes called the **modified midpoint method**.[151])

Note the distinction between **steps** and **passes** for Gragg and midpoint. A Gragg 6-step calculation does 7 passes. The LOG file will refer to pass 3 of a 7-pass calculation.

Our experience is that, provided your simulation is not too nonlinear,

- Gragg or midpoint will converge significantly faster than Euler (that is, they produce more accurate results for the same number of steps), and
- the extra accuracy Gragg usually gets from the extra pass it does compared to the midpoint method is usually well worth the extra time taken.

However, we have found that in some highly nonlinear simulations, Gragg and midpoint diverge rapidly (see section 12.6.5 for an example). If this happens, the Extrapolation Accuracy Summary will tell you clearly, and you should then try Euler's method (though that may not converge either in these cases). More information about this issue is given in sections 12.6.3 and 12.6.4 below.

Note that, if you are using Gragg's method or the midpoint method and are extrapolating from two or three separate calculations, all step numbers must be even or all must be odd. (Thus extrapolating from 2,3,4-step Gragg is not allowed.) We recommend 2,4,6-step as the simplest Gragg or midpoint. (A 1-step Gragg or midpoint is really Euler, so doesn't make much sense.)

Because Gragg's method takes the exogenous variables past their simulation end-point on the final pass, it may not be suitable for some simulations of models in which it is not sensible to increase the exogenous variables in this manner. In such a case we recommend the midpoint method.

Using any of the three methods, if a percentage change variable is shocked by more than -100 per cent, this may leave its levels value dangerously close to zero after some step in the simulation. In this case GEMSIM or the TABLO-generated program will tell you that you must change the shock or the number of steps.

If you need to give a **percentage change shock of exactly –100 percent** (which is necessary if you want to reduce a positive value to zero in the levels), you will not be able to use Gragg's method since the levels value of this variable would be zero at the start of the final (N+1)th pass (see above). However you can use the **midpoint** method in this case. [This is why we have provided the midpoint method.]

12.3 Reusing Pivots

For large models, the time taken for the LU decomposition (see section 12.1) of the matrix[152] A(K) may be significant. (Indeed, the time taken for it may exceed the time taken for all other parts of the current step.) In this connection you should be aware that, in steps 2,3 and so on of a multi-step simulation, the Harwell sparse matrix routines can calculate the LU decomposition significantly quicker if they can make use of the position of the so-called **pivots** calculated when LU decomposing the corresponding matrix A(1) in step 1. This is called **reusing pivots**. If successful, the time taken for the LU decomposition in steps 2,3 etc is often less than half that for step 1.[153]

---

[151] Euler, midpoint and Gragg are well-known methods for solving initial value problems – see, for example, Chapter 15 of Press et al (1986) or Chapter 6 of Atkinson (1989).

[152] This is the matrix called A(K) in section 6.2. This matrix changes from step to step as the data is progressively updated.

[153] The Harwell routine MA28A always does the LU decomposition at step 1. The Harwell routine MA28B is the one invoked at steps 2,3 etc; it attempts to use the pivot positions as found by MA28A. If it is unsuccessful (which can happen for a variety of reasons), MA28A is called to calculate the LU decomposition from scratch.

However occasionally reusing pivots can reduce the numerical accuracy of solutions slightly (though this should not happen if iterative refinement is used). If you are in doubt about this, you can prohibit reuse of pivots by including the statement[154]

```
NRP = yes ;  ! No Reuse of Pivots    [Default is "NO" which means "reuse pivots".]
```

in your Command file.

You may also prefer to prohibit the reuse of pivots if experience shows that reuse of pivots is failing with a particular model or closure; then including "nrp = yes ;" in your Command file will save the time spent attempting unsuccessfully to reuse pivots.

In our experience, reusing pivots does not reduce the simulation time as much when MA48 is being used as when MA28 is being used. Reuse of pivots also requires large, temporary work files, especially with MA48. If you use MA48, you might like to experiment to see if reusing pivots is reducing simulation time or not. [We will be interested to hear the results.]

12.4 Time Taken for the Different Steps

You will notice that steps 2,3 etc take essentially the same time as each other. However the time taken for step 1 is usually different from the time for steps 2,3 etc, as explained below.

Steps 2,3 etc Are Usually Quicker Than Step 1

Usually the time taken for steps 2,3 etc is less than that for step 1 for four reasons.

- Reusing pivots (see section 12.3) usually makes a considerable reduction in the time taken.
- DISPLAYs and WRITEs are usually only done during step 1.
- READs are not usually necessary at steps 2,3 etc since the updated values are usually in memory.
- By step 2, the software knows the closure. It then does not need to calculate any submatrices corresponding to exogenous variables which are not shocked.

If Extrapolating, Step 1 is Very Quick for Solutions Two and Three

If you are doing two or three multi-step calculations and extrapolating from them, you will notice that relatively little time is taken for step 1 of the second and third solutions. There are two reasons.

- The formulas and equations do not need to be recalculated. This is because the starting point for step 1 is the initial data base. During step 1 of the first solution, GEMSIM or the TABLO-generated program has stored on the Base Coefficient Values file (see section 9.2) the values of all coefficients (calculated from this base data) needed for backsolving and updating. Thus, in step 1 of the second and third solutions, GEMSIM or the TABLO-generated program does not need to calculate any formulas; instead it reads the values it needs from the Base Coefficient Values files.
- The LU decomposition does not need to be recalculated since the solutions for the endogenous variables (in the condensed system) for this step have already been calculated during step one of the first multi-step calculation. Hence the submatrices do not need to be recalculated either.[155]

Thus the time taken for step 1 of the second and third solutions is essentially that for doing the backsolves (if any) and the updates.

12.4.1 Total Time for Multi-step Compared to 1-step

If you have carried out a 1-step simulation including updates, you can form a rough estimate of the time that will be required to calculate one or more multi-step solutions.

---

[154] Alternatively, select option NRP (see section 14.1).

[155] The LHS Matrix (see section 2.13.2 of GPD-1) is the same in step 1 for all 3 calculations. During step 1 of the first calculation, the program has calculated all 3 right-hand side vectors and solved all 3 systems of linear equations [as in equation (3) in section 2.13.2 of GPD-1], using the LU decomposition of the LHS Matrix A.

The total time taken for a single N-step simulation should be no more than

- N times that for a 1-step if you are using Euler's method or the midpoint method;
- (N+1) times that for a 1-step if you are using Gragg's method. (This is because an N-step Gragg actually makes (N+1) passes.)

In fact the time for an N-step should be less than these estimates since steps 2,3 etc usually take less time than step 1 (as explained in section 12.4 above).

If you extrapolate on the basis of two or more multi-step solutions, you can pretty well ignore the time taken for step 1 of the second and third solutions (as explained in section 12.4 above). Also the time taken for the extrapolation (after the 2 or 3 solutions have been calculated) is usually negligible compared to the rest. Thus,

- the time taken for simulation plus extrapolation from N1 and N2-step solutions is roughly the time for an N1-step solution plus time for an (N2-1)-step solution.
- the time taken for simulation plus extrapolation from N1,N2 and N3-step solutions is roughly the sum of the times for separate N1, (N2-1) and (N3-1) step solutions.

Thus, for example, to a first order of approximation,

1. extrapolating on the basis of 2,4,6-step Euler solutions is likely to take no more than 2+3+5=10 times as long as a Johansen simulation (including update) while
2. extrapolating on the basis of 2,4,6-step Gragg solutions is likely to be no more than for 2+4+6=12 times as long as a Johansen simulation.

You can get even better estimates on the basis of a single 2-step simulation. Suppose step 1 takes T1 seconds and step 2 takes T2 seconds. Then, since the times for each of steps 2,3 etc is essentially the same,

- the time taken for a single N-step Euler or midpoint solution will be approximately

  T1 + (N-1).T2 seconds.

- the time taken for a single N-step (but N+1 pass) Gragg solution will be approximately

  T1 + N.T2 seconds.

12.4.2 Using Existing Files to Speed Up Step 1

When you run GEMSIM or a TABLO-generated program, the Equations file (see section 9.1) produced contains all coefficients of the equations as evaluated from the starting data, while the Base Coefficient Values file (see section 9.2) contains the values of all coefficients needed for the backsolves and updates, also based on the starting data. If you run a second simulation with GEMSIM or the same TABLO-generated program and are starting from the same data, you may be able to speed up the calculation of step 1 by telling the program to

use the existing Equations and Base Coefficient Values files.

This will reduce the time for step 1 since the formulas and equations don't need to be recalculated.

Of course, if you change the theory (that is, the TABLO Input file) or the data, the existing Equations and Base Coefficient Values files will still reflect the old theory and base data and so it would not be appropriate to reuse them.

If you have already carried out a simulation based on the same theory, data and closure, you can eliminate the need to compute the LU decomposition at step 1 by telling GEMSIM or the TABLO-generated program to

use the relevant LU file

(see section 9.3) provided you saved it on the previous run. This may further speed up step 1. Note that you can only use an LU file if you are also starting from existing Equations and BCV files.

However

we no longer recommend the use of existing Equations and BCV files (or of LU files)

for the reasons set out in section 9.2.1. With the relatively fast machines available today, the time savings are not very important in most cases. Any time saving must be traded off against the loss of flexibility (since various features as listed in section 9.2.1 are not compatible with starting from existing Equations and BCV

files) and the possibility of error (for the reasons listed below). As indicated in section 9.2.1, we are seriously considering not supporting this feature in the next version of GEMPACK.

If you do use existing Equations and BCV files, you should note that

- Equations and Base Coefficient Values files depend on the theory (that is, the TABLO Input file) and the base data. You should only use existing Equations and Base Coefficient Values files which are based on the same theory and base data as you intend to use for the current simulation.
- LU files depend on the theory, base data and closure. You should only use an existing LU file if it is obtained using the same theory, base data and closure you intend to use for the current simulation.

The Equations file produced by GEMSIM or a TABLO-generated program is a suitable starting point for Johansen simulations via SAGEM. Also an LU file created by SAGEM can be used with GEMSIM or a TABLO-generated program (or vice versa) provided the theory, data and closure are the same.

12.5 Ignoring / Keeping Zero Coefficients

GEMPACK is able to solve large systems of linear equations efficiently by exploiting the sparsity of the LHS matrices of coefficients. The Harwell sparse matrix routines (see section 12.1.1) used by GEMPACK usually only need to be told about the nonzero entries in these matrices.

Consider, for example, the "Com_clear" equations for Stylized Johansen as shown in section 3.3.3 of GPD-1. The linearized version of these are

```
(all,i,SECT)  XCOM(i)/100.0*p_XCOM(i) - {XH(i)/100.0*p_XH(i)
                + SUM(j,SECT, XC(i,j)/100.0*p_XC(i,j) ) } = 0
```

In the tableau for the Equations Matrix C (as set out in Table 2.13.1 of GPD-1), notice that these two equations occupy rows 11 and 12 (see the SUMEQ map in section 13.1.1 of GPD-4). Since only variables p_XCOM, p_XH and p_XC occur in these equations, all entries in the other columns of these rows in the tableau are necessarily zero (irrespective of the data base values). The first of these two equations (the one referring to sector "s1") can only have nonzero entries in the columns corresponding to variable components

p_XCOM("s1"), p_XH("s1"), p_XC("s1","s1") and p_XC("s1","s2").

(These are columns 6, 10, 12 and 14 respectively, as shown in the SUMEQ map in section 13.1.1 of GPD-4.) The entry in the column corresponding to p_XH("s1") is -XH("s1")/100.0 which is -2/100.0 for the usual data base (as shown in Table 2.1.1a of GPD-1). If however households consume none of commodity 1 (so that DVHOUS("s1"), and hence XH("s1)", are zero in the data base), this entry would be zero. This is what we mean by a **zero coefficient** - namely one that is zero because of the data base currently being used **but might possibly be nonzero given another data base**. (Contrast this to the coefficient of variable 'p_Y' in these equations which will always be zero, irrespective of the data base, since variable 'p_Y' does not appear in these equations.)

If you keep zero coefficients (in the sense just defined) during step 1 of a multi-step simulation, this maximises the chance that reusing pivots (see section 12.3 above) will succeed in steps 2,3 etc. (Since the data base changes at these steps because of updating, a zero coefficient at step 1 may possibly become nonzero in later steps.) Reuse of pivots fails if the LHS Matrix at step 2 has an entry in a position for which no entry was recorded at step 1.) However keeping zero coefficients calculated at step 1 may increase significantly the time taken for step 1.

For many models, reusing pivots will succeed at all subsequent steps even if zero coefficients are not kept during step 1.

For some models and some simulations, reusing pivots fails unless zero coefficients are kept during step 1. In such cases, in a multi-step simulation with several steps, the extra time taken during step a may be more than offset by the speedup in later steps achieved by reusing pivots. (Though this would not be a consideration for a 1-step simulation and may not be for simulations with a small number of steps.)

To give you control over this, we have provided the Command file statement

IZ1 = YES|no ; ! Default is YES. Put "no" if you do not want to ignore zero coefficients at step 1

which tells GEMSIM and TABLO-generated programs whether or not to ignore zero coefficients at step 1 ["yes" says to ignore them, so "no" means to keep them]. Because we have found that keeping zero coefficients

in step 1 only speeds up simulations with many steps and only for some models, "IZ1 = yes ;" is the default. If you wish to keep zero coefficients at step 1, you can include statement "IZ1 = no ;" in your Command file.[156]

At steps 2,3 etc, recording (or keeping) zero coefficients is usually counter-productive (the pivots the program attempts to reuse at step 3 are usually those from step 1, not from step 2) and so the default is not to record zero coefficients at steps 2,3 etc. However, in a few cases we have found that, paradoxically, the LU decomposition proceeds faster if zero coefficients are kept. Hence we have provided Command file statement

KZ2 = yes|NO ; ! default is NO. Put "yes" if you wish to keep zero coefficients at steps 2,3 etc

which tells GEMSIM and TABLO-generated programs whether or not to keep zero coefficients at steps 2,3 etc["yes" says to keep them and "no" says to ignore them]. If reuse of pivots is failing, you may like to experiment with including the statement "KZ2 = yes ;"[157] in your Command file to see if it speeds up calculations; however it will usually slow things down, and will almost certainly not be an improvement if reusing pivots is succeeding.

These options about IZ1 and KZ2 may be useful if you have a singular LHS Matrix – see section 15.1 for details.

12.6 Equation Solving Problems

12.6.1 Warnings About Equations Not Being Satisfied Very Accurately

After solving a system of equations

$$A\,y = b$$

(as in equation (3) of section 2.13.2 in GPD-1), GEMSIM or the TABLO-generated program (or SAGEM) usually calculates the product Ay (using the solution values for y) and compares it entry by entry with the RHS vector b. If these two appear to differ significantly, the program reports the equation numbers where these differences occur. The report will look something like the following.[158]

```
%%WARNING. Equation MarketClear("food","USA") is not satisfied very accurately.
(Sum of its terms is 0.2340 while sum of their absolute values is 2.230000.)
This may be because the LHS matrix is not really invertible.
```

If GEMSIM or your TABLO-generated program (or SAGEM) gives one or more of these warnings about equations not being satisfied very accurately, you should check the equation or equations indicated in the messages.[159]

---

[156] Option IZ1 (see section 14.1) relates to this. By default, this option IZ1 is selected (since "IZ1 = yes ;" is the default). If you turn that off (by entering response **-IZ1** at the options screen, that will have the same effect as including the statement "IZ1 = no ;" in your Command file. ["IZ1 = no ;" was the default in Release 5.0. The default was changed to "IZ1 = yes ;" in Release 5.1.]

[157] Alternatively you can select option KZ2 (see section 14.1).

[158] Prior to Release 7.0, these messages only indicated the equation number, not name. In Release 7.0, TABLO-generated programs and GEMSIM now indicate the equation name and arguments (though SAGEM still just gives the equation number).

[159] Consider the equation C1*x1 + C2*x2 = C3*x3. This is rewritten as C1*x1 + C2*x2 – C3*x3 = 0 before solving. In this case the "sum of the terms" referred to in the warning message indicates the result of evaluating the LHS of the rewritten equation using the alleged solution values for the variables. This should always be equal to zero if the equation is satisfied exactly. The "sum of their absolute values" in this case is the sum of the absolute values of the terms C1*x1, C2*x2 and C3*x3. You can see how far from being satisfied accurately the equation is by comparing the size of the "sum of the terms" with the "sum of their absolute values".

You should also be suspicious of your simulation results since these messages often indicate some serious problem with the equations of your model or the closure you are using. Accordingly you should check the alleged solution carefully since it may be unreliable (or even meaningless).

Indeed, as the message says, such warnings often mean that the LHS matrix is not really invertible.[160] You should be aware that, in general, it is impossible to accurately distinguish numerically between a singular LHS Matrix (that is, one that is not invertible) and one that is nearly singular. (This is because of the limited precision of computers and the rounding errors that necessarily occur when doing large numerical computations.) In modelling, singular matrices are usually an indication that an invalid closure is being used, though occasionally they can be a consequence of zeros in the data base. See section 15.1 for details about singular matrices.

Experienced GEMPACK users tend to ignore warnings since a large number are produced in many cases. However,

we urge all users to take VERY SERIOUSLY these warnings

about equations not being satisfied very accurately. Indeed, if you receive a large number of these messages, the program runs to completion, saves all files, but then ends with a fatal error to alert you to the seriousness of the warnings.[161]

If any of these warnings is given during a simulation, a message such as the following will be given near the end of your LOG file.

```
%%Warning. There have been 175 warnings about equations not being satisfied very
accurately. The more of these there have been, the more likely it is that the
solution is not valid, indeed that one or more of the matrices is not really
invertible.
```

The other way of checking if you have received any of these warnings is to search for

not satisfied very accurately

in your LOG file.

If, despite the above, you wish to suppress these warnings, you can do so by adding the statement[162]

NWE = yes ; ! No Warnings about Equations not being satisfied

in your Command file. [The default is "NWE = no ;" which means "give warnings".]

12.6.2 Warnings About Variation Between Passes in Endogenous Results

If your simulation is a highly non-linear one, you may see warnings of the following kind.

```
 %%WARNING. Variation in endogenous values between this
  and the previous pass may indicate nonconvergence of the solutions.
 Sum of absolute values of the endogenous variables on pass 2 is 23456.0,
  while this sum was 78432.1 on the previous pass.
```

This message indicates a significant difference in the size of the endogenous results from one pass to the next one. However, by itself,

this does not mean that your simulation is not converging satisfactorily.

---

[160] Alternatively they may indicate that the LHS matrix is **ill-conditioned** [see, for example, section 8.4 of Atkinson (1989)]. This means that a very small change in the values of the exogenous variables might result in a disproportionately large change in the values of some of the endogenous results.

[161] Ending with a fatal error was introduced in Release 6.0. This happens if more than 100 such warnings are given.

[162] Alternatively, select option NWE (see section 14.1).

Convergence

- **is indicated best** by the Extrapolation Accuracy Summaries (see section 7.2) or,
- if in doubt, by the results on the Extrapolation Accuracy file (section 7.2.3).

When you receive messages about variation between passes, you should check this Extrapolation Accuracy information carefully. If in doubt, perhaps increase the number of steps and/or subintervals used in solving the model (or use automatic accuracy). See sections 7.1 and 7.1.1 for advice about increasing accuracy.

12.6.3 Gragg and Midpoint Are Not Suitable in Some Cases

Gragg's method or the midpoint method converge much more quickly than Euler's method for many simulations. (That is, they produce much more accurate results for the same number of steps.)

However these methods are known to be unsuitable for some simulations because the different multi-step simulations (for example, 6-step, 8-step and 10-step) may oscillate rather than converging monotonically as we would like. In such cases, it is known that these oscillations may become worse when the number of steps in increased. You can obtain information about oscillations from the Extrapolation Accuracy file and Summary (see section 7.2).

It is virtually impossible to tell in advance which simulations will not converge with Gragg or the midpoint methods. Changing closure and especially shocks can and will change the behaviour. The practical moral seems to be the following.

> **If you see increasingly large oscillations in your results when using Gragg or the midpoint method, switch to Euler's method.**

Euler's method does not suffer from the problem of increasingly large oscillations.[163]

A specific example in which there are increasingly large oscillations is given in section 12.6.5 below.

12.6.4 Linearization Used Can Affect Accuracy

We begin this subsection with an example, and conclude with some general comments.

Example

Consider the following levels TABLO Input file for the equation D=P*Q (Dollar value equals Price times Quantity).

```
VARIABLE (LEVELS) D  # Dollar value # ;
VARIABLE (LEVELS) P  # Price # ;  VARIABLE (LEVELS) Q  # Quantity # ;
FORMULA (INITIAL)  P = 1 ;    FORMULA (INITIAL)  Q = 1 ;
FORMULA & EQUATION  Value     D = P*Q ;
```

Example TABLO Input file

We suppose, for the purposes of this example, that Q is a stock whose levels value can be either positive or negative.

Consider the simulation in which P and Q are taken from their initial levels values of 1 and 1 to 1.1 and -0.18 respectively. This involves shocking the associated linearized variables p_P and p_Q by 10 and -118 per cent respectively. The new (post-simulation) levels value of D should be

D = 1.1*(-0.18) = -0.198

[163] Technically, Gragg's method and the midpoint method are only what is called "weakly stable". See, for example, section 6.4 of Atkinson (1989).

so that the exact simulation result is p_D = -119.8 (that is, a 119.8 per cent decrease in D).

You can run TABLO and then GEMSIM to carry out this simulation using different step numbers and solution methods. You can also see the effect of two different linearizations by

1. running TABLO with the default CHECK options, or
2. running TABLO after selecting option ACD ("Always use Change Differentiation of levels equations") at the CHECK stage (see section 2.2.6 of GPD-2).

For the equation D=P*Q, these produce two different linearizations, namely

1. p_D = p_P + p_Q

2. D*p_D = P*Q*[p_P + p_Q]

(as you can see by looking in the Information file, as described in section 9.3 of GPD-2). The first of these comes from applying the Percentage Change rules for differentiating products (see section 9.1 of GPD-2) while the second comes from applying the Change differentiation rule for products (also in section 9.1 of GPD-2).

The numerical results for this simulation are surprisingly different for these two linearizations. For example, for 6,8,10-step Euler calculations, the results for p_D are

```
                          6-step    8-step    10-step   extrap
(a) Percent Change diff  -123.734  -119.226  -120.106  -150.510 OC? 2
(b) Change diff          -117.833  -118.325  -118.620  -119.799 XMA 6
```

As you can see, Change differentiation produces a very accurate extrapolated result, whereas Percentage Change differentiation produces poor results. If you experiment with different numbers of steps and/or different solution methods, you will see the same pattern.

General Comments About the Different Linearizations

Our experience to date with different models indicates that better numerical convergence is sometimes obtained when Change differentiation is used throughout. It is for this reason that we have made the **ACD** option (see section 2.2.6 of GPD-2) available in TABLO.

If you are experiencing convergence problems with a model containing levels equations we recommend that you try selecting ACD to see if this helps. If your model is a linearized one, you will need to carry out the alternative linearizations by hand, which will be more difficult.

Note that it is only for some simulations that there is a significant difference between the different linearizations. With the D=P*Q example above, if you change the simulation to one in which Q is reduced only by 95 per cent (rather than 118 per cent) you will find that both the Percent Change and Change differentiations result in acceptable convergence. For example, the 6,8,10-step Euler results are shown below. (The exact result is p_D = -94.5.)

```
                          6-step    8-step    10-step   extrap
(a) Percent Change diff  -94.0610  -94.2100  -94.2860  -94.4887 CX 2
(b) Change diff          -92.9167  -93.3125  -93.5500  -94.4995 CX 5
```

12.6.5 Example of Increasingly Large Oscillations

Consider the Percentage Change differentiation version of the D=P*Q example in section 12.6.4 above in which the shocks are 10 to p_P and –118 to p_Q.

This is an example of the type of problem mentioned in section 12.6.3 above in which Gragg and/or midpoint produce increasingly large oscillations as the number of steps increases.

For example, some different Gragg results for p_D are:

```
  4-step      6-step      8-step     10-step    14-step    20-step
 -130.630    -154.549    -105.889   -110.894   -141.567   -63.4437
```

Even without knowing the correct result, you can see that the oscillations are getting worse as the number of steps increases in this case.[164] See section 12.6.3 above for advice if you encounter this in your model.[165]

12.6.6 Scaling Equations

It is possible to ask the software (including SAGEM) to scale the equations via the Command file statement

```
scale equations = yes|NO ;    ! default is no
```

We have found that automatic scaling of the equations (done by routines provided by Harwell with MA28 and MA48) greatly improves the accuracy with which the linear equations are solved in a few cases. The model which convinced us to include this as an option was a levels macroeconometric model (which possibly has rather different numerical properties from the AGE models GEMPACK is often used for).

But our testing with other "standard" GE models (including ORANI's and GTAP) have indicated that this scaling does not help and, in some cases, produces less accurate results. So we **do not recommend its wide-spread use.**

12.7 Work Files

GEMSIM and TABLO-generated programs often produce large, temporary work files when they are carrying out a simulation. These files are usually deleted when the run is finished. However, if the program crashes, sometimes these work files are left on your hard disk.

Several of these work files have suffixes ending in WK, including **.CWK**, **.DWK**, **.PWK** and **.SWK**. Other work file suffixes are **.E2K**, **.EXS** and **.EL2**. It is always safe to delete such work files if you see them lying around.

Sometimes a simulation will crash with an error message suggesting that your disk is full. Usually the program cleans up the work files after such a message. This explains why you may be puzzled to find 100Mb or more free after such a message. Even if you find such a large amount of free disk space, you should try to free up more after such an error message.

The work files produced by GEMSIM or a TABLO-generated program may take up **several hundred megabytes** of disk space. In general, the larger your model, the larger these work files will be.

---

[164] The extrapolated results with Gragg and Percentage Change differentiation are even worse. For example, the extrapolated result from the Gragg 4,6,8-step results is 0.127 while the extrapolated result from the Gragg 10,14,20-step results is 73.327. The correct result is p_D = -119.8 (see section 12.6.4).

[165] If you encounter an example of increasingly large oscillations in one of the standard GE models, we would be interested in hearing the details.

CHAPTER 13

13. Memory Management

All GEMPACK programs are compiled with a suitable Fortran 90 compiler (see section 5.6 of GPD-1).

In this chapter we describe aspects of memory allocation and management for GEMSIM, TABLO-generated programs and SAGEM which you may need to understand in order to ensure that these programs carry out simulations as quickly as possible.

13.1 Automatic Memory Allocation

Because they are compiled with a Fortran 90 compiler, the GEMPACK programs are able to use the memory management features of Fortran 90 to allocate just enough memory to complete each task. [If there is not sufficient memory available on your computer, the programs will report this and stop.[166]]

This memory allocation is done at run time when the size of the relevant arrays are known. There is no need to recompile the program after increasing the size of your model.[167]

Because of the more efficient memory use, the programs are quicker to load. Because they just allocate just what is necessary for each task, the tighter array limits mean that the program is more likely to fit within the memory available. When GEMPACK is installed on a network, the same GEMPACK program can be run on machines of small and large memory sizes, and larger machines can still run large problems without the need to adjust program parameters.

Apart from the MAXIMUM SIZE change described in section 4.6.2 of GPD-2, the change from Fortran 77 to Fortran 90 memory management should be transparent to users. You will not see anything different; rather it is the absence of those annoying "increase parameter" messages (which were a result of Fortran 77 memory management) which is the difference.

13.2 Preliminary Pass

When GEMSIM or a TABLO-generated program run, they perform what they refer to as a "preliminary pass" in which they calculate the sizes of all sets and compute all subset mappings. On this pass they only do whatever is required to get this information. [For example, they only do FORMULAs if some set is dependent on the values of a particular COEFFICIENT (see section 4.6.5 of GPD-2).]

[166] The message will say that the program is unable to allocate sufficient memory.

[167] Users familiar with earlier versions of GEMPACK based on a Fortran 77 compiler will find that they no longer get messages asking them to increase program parameters.

13.3 Repetitions if Memory Allocation is Too Small

You may see GEMSIM and TABLO-generated programs redoing the submatrices a couple of times before they allocate sufficient memory to carry out the LU decomposition. This repetition does not usually take more than a few seconds (and is certainly quicker than recompiling as was necessary with Release 5.2).

SAGEM may also read the Equations file a couple of times before it allocates enough memory for the LU decomposition.

However, if you wish to help the programs out, you can put one of the statements

```
start with MMNZ|MMNZ1 = <integer value> ;
                ! Sets initial memory allocation
                ! for nonzeros in GEMSIM, TABLO-generated programs or SAGEM
```

into your Command file. For example, if you are using MA48, just set MMNZ via a statement like

```
start with MMNZ = 50000 ;     ! [This sets MMNZ1 to the same value.]
```

If you are using MA28 you may want to set MMNZ and MMNZ1 separately.

Whenever GEMSIM, a TABLO-generated program or SAGEM completes a simulation, it tells you the minimum value for MMNZ and MMNZ1 that would have worked. But if you use the statements above to set initial values of MMNZ and/or MMNZ1, we suggest that you set them a few percent higher than these minimum values otherwise the solution may take longer than necessary.

Note that, if you use the statements above and set MMNZ or MMNZ1 too low, the programs will still continue and find appropriate values.

The "start with MMNZ|MMNZ1 = … ;" statements are also available with SAGEM.

Example – GTAP Command file GC10E1.CMF

Look in the Command file GC10E1.CMF (see also section 6.4.3) supplied with the GEMPACK examples. Note that this contains the statement

```
Start with MMNZ = 210000 ;
```

To see the effect of this statement, comment it out and run the experiment. Search in the resulting LOG file for "Redoing submatrices". The program only knows an appropriate value for MMNZ after it tries to calculate the LU decomposition the first time. It then needs to redo all the submatrices with this value of MMNZ. Now reinstate the above statement and run the experiment. Again search the LOG file for "Redoing submatrices". It will not be present in the LOG file this time.

Note that it is virtually impossible to guess in advance the relevant value for MMNZ. In practice, you need to run a simulation with the relevant model, data and closure once. The minimum suitable value for MMNZ is shown in near the end of the LOG file. Then you can insert appropriate "start with MMNZ" statements in this and other Command files with the same model, data and closure.

Note that GEMSIM and TABLO-generated programs can also efficiently increase the size of MMNZ and/or MMNZ1 if necessary during any step of a multi-step calculation.

Windows programs such as RunGEM, RunDynam and RunMONASH (see chapter 2 of GPD-4) write Command files based on information provided on various pages of their notebooks. They never make a guess about appropriate values of MMNZ or MMNZ1. If you wish to specify starting values for these (which may make the programs run noticeably more quickly), you can include the relevant statement (for example, "start with mmnz = 100000 ;") in a file called **CMFSTART** (no suffix) in the relevant directory. To find out about this, click on ***File | Edit CMFSTART file*** in the main menu and then click on the Help button.

13.4 Reporting Memory Used by TABLO-generated Programs and GEMSIM

TABLO-generated programs and GEMSIM give reports as to how much memory is required (or used).[168]

The memory used is reported in two parts (referred to as the TGMEM1 and TGMEM2 parts). For large models, the TGMEM1 part is usually the larger of these.

Note that these reports do not include memory required for the nonzeros generated doing the LU decomposition (where memory is allocated according to the size of the parameters MMNZ and MMNZ1). The amount of memory used for these is

12*MMNZ bytes	if MA48 is used,
(4*MMNZ + 8*MMNZ1) bytes	if MA28 is used.

The minimum MMNZ and MMNZ1 values required are echoed at the end of the run of these programs.

With these reports you can get a rough estimate of the total memory required to run the program. To do this, add the TGMEM1 and TGMEM2 parts, add the above values for MMNZ,MMNZ1 and then add about 4-5 megabytes[169] for other memory and for memory required for the code itself.

[168] Memory reports were introduced in Release 6.0-001.

[169] For a large model such as MONASH or GTEM, you may have to increase the 4-5 Mbytes to 10 Mbytes.

CHAPTER 14

14. Options for GEMSIM and TABLO-generated Programs

GEMSIM and TABLO-generated programs offer several different options, which can affect the way they run (for example, how they carry out a simulation).

The options screen for TABLO-generated programs is shown below.[170]

```
                TABLO-GENERATED PROGRAM OPTIONS
             ( --> indicates those in effect )

BAT Run in batch           STI Take inputs from a Stored-input file
BPR Brief prompts          SIF Store inputs on a file
LOG Output to log file     ASI Add to incomplete Stored-input file
  NAX Name AXS/T files     CMF Take inputs from a Command file
EAA Echo all activity        SVX Select variables on XAC file (later)
CPU Report CPU times         RQF Required figures agreement (extrap)
NRP Don't reuse pivots     NIR Don't do solution iterative refinement
                         -->IZ1 Ignore zero coefficients in step 1
                           KZ2 Keep zero coefficients in steps 2,3 etc
NEQ Do no equations        NWE Don't warn how well equations solved
NDS Do no displays         SSI Several subintervals; extrap after each
NWR Do no writes           M28 Use MA28 routines (rather than MA48)
NAS Do no assertions       SUI Save updated formula(initial) data
NUD Do no final updates           Extra Options Screen
NSM Don't do simulation           --------------------
NWT Use Newton's method    DTO Display, Terminal/Text write Options
CR  Check-on-read options

Select an option   :  <opt>  Deselect an option      : -<opt>
Help for an option : ?<opt>  Help on all options     : ??
Redisplay options  : /       Finish option selection : Carriage return

Your selection >
```

Options Screen for TABLO-generated Programs

The options screen for GEMSIM is the same except that the option NAX, which is not relevant for GEMSIM, is not shown.

Note that, in both cases, option IZ1 is selected by default[171] (though you can, of course, turn it off).

As indicated in section 5.4.2 of GPD-1, Online Help screens can be displayed while running these programs by typing '**?**' followed by the abbreviation to the option on which help is needed. For example, to obtain help on Log files, type **?LOG**. To obtain help on all options, type **??** and scroll through the screens using a carriage-return. However, unless you are doing something special, you should probably

use the default options plus the option CMF.

---

[170] The odd-looking indenting in this menu is an attempt to show the groups of related options. We would have preferred to have left blank lines to show these, but unfortunately this would have made the menu too long for most screens.

[171] The reason is explained in section 12.5.

For this reason, you may prefer to skip chapter 14 initially and only return to it later if and when you feel you need the more specialised information and advice given below. Note that most of the options described below can be activated by appropriate statements in GEMPACK Command files; we mention the related Command file statements when discussing the options below.

Option **NAX** for TABLO-generated programs allows you to specify the names of the Auxiliary files, and corresponds to the Command file statement "Auxiliary files = … ;". As indicated in section 3.2, this Command file statement (and hence this option NAX) is only relevant in special cases (see sections 3.2.1 and 3.2.2).

14.1 Options Affecting Simulations

Several of the options available with GEMSIM and TABLO-generated programs affect the way simulation results are calculated. Some of these are aimed at possibly speeding up the calculations.

The relevant options (with their Command file equivalents also shown) are:

IZ1 Ignore zero coefficients in step 1 ("IZ1 = yes ;" in Command files)

KZ2 Keep zero coefficients in steps 2,3 etc ("KZ2 = yes ;" in Command files)

NRP Don't reuse pivots ("NRP = yes ;" in Command files)

NIR Don't do solution iterative refinement ("NIR = yes ;" in Command files)

NWE Don't warn how well equations solved ("NWE = yes ;" in Command files)

NWT Use Newton's method ("NWT = yes ;" in Command files)

SSI Several subintervals: extrap after each ("subintervals = <number> ;" in Command files)

The effects of IZ1 and KZ2 are described in section 12.5. The effects of NRP, NIR and NWE are described in sections 12.3, 12.1 and 12.6.1 respectively. The effect of NWT is described in section 7.5. The effect of SSI (several subintervals) is described in section 7.3.

14.2 Options Affecting Extrapolation Accuracy Summaries and Files

The relevant options (with their Command file equivalents also shown) are:

RQF Required figures agreement (extrap) ("RQF = <integer> ;" in Command files)

SVX Select variables on XAC file (later) ("xac-retained endogenous = <list> ;" in Command files)

These are described in section 7.2.4.

14.3 Saving Updated Values from All FORMULA(INITIAL)s

An option

SUI **S**ave **U**pdated values from formula **I**nitials ("sui = yes ;" in Command files)

in GEMSIM and TABLO-generated programs allows you to save updated values of all coefficients whose initial values are set via FORMULA(INITIAL)s. See section 7.7.2 for details.

14.4 Options Affecting the Actions Carried Out

These can be used to suppress actions that GEMSIM or the TABLO-generated program is capable of carrying out. The relevant options are shown below. Each has a Command file equivalent (which you should prefer to use as a general principle) which is also shown.

NEQ Do no equations ("NEQ = yes ;" in Command files)

NDS Do no displays ("NDS = yes ;" in Command files)

NWR Do no writes ("NWR = yes ;" in Command files)

NUD Do no final updates ("NUD = yes ;" in Command files)

NAS Do no assertions ("assertions = no ;" in Command files)

NSM Don't do simulation ("simulation = no ;" in Command files)

The effects of **NDS, NWR, NAS, NUD, NEQ** and **NSM** are described in section 6.1.7 above.

Note that the options NSC and NSE, which were available in Release 6.0 and earlier, and which affected saving Equations and BCV files, are no longer available. See sections 9.1 and 9.2 if you want to save an Equations file or a BCV file.

14.5 Options Affecting How Writes and Displays are Carried Out

There are several options affecting how writes to text files and displays are carried out. If you are running interactively or via a Stored-input file, you can see these options by first selecting option

DTO Display, Terminal/Text write Options

This reveals the following menu.

```
               DISPLAY/TERMINAL/TEXT FILE OPTIONS
               ( --> indicates those in effect )

      DWS Displays, terminal writes at all steps of a multi-step sim
      TWC Terminal writes in column order
      DPW Change page width of display file
      DPL Change page length of display file
      DDC Change number of figures after decimal point in displays
 -->  DPN New page in display files only if needed
      D1C In display files, show 1-dimensional array as a column
      DOI In display files, omit "identical" row messages
      NEL No element labels on ROW_ORDER or COL_ORDER text files

Select an option   :  <opt>  Deselect an option      : -<opt>
Help for an option : ?<opt>  Help on all options     : ??
Redisplay options  : /       Return to other Options : Carriage return

 Your selection >
```

DTO Display, Terminal/Text Write Options Menu

Once you have selected any options desired from this, you return to the main options menu by entering a carriage-return.

Alternatively, if you are using a GEMPACK Command file, you can select these options directly - option **DTO** is not relevant. For example, "display width = 70 ;" sets the width of pages in the display file to 70.

Options DWS and TWC are equivalent to the Command file statements "dws = yes ;" (see section 6.1.9) and "twc = yes ;" (see section 6.1.11).

Display files are introduced in section 4.3. Options DPW, DPL, DDC, DPN, D1C and DOI are the analogues of the respective Command file statements described in section 4.3.

Normally element name or number labels are shown (as comments) in row_order or col_order text data files (see section 6.2.14 of GPD-4). If you select the option

NEL No element labels on ROW_ORDER or COL_ORDER text files

or put the statement "**NEL = yes ;**" in your Command file, such labels will be suppressed. (See chapter 6 of GPD-4 for more information about these GEMPACK text data files.)

14.6 Options Affecting CPU and Activity Reporting

The relevant options (with their Command file equivalents also shown) are:

CPU Report CPU times ("CPU = yes ;" in Command files)

EAA Echo All Activity ("EAA = yes ;" in Command files)

These options are described in sections 2.6 and 6.1.10 respectively.

14.7 Special Options for SAGEM

Several options are available for SAGEM apart from the basic options BAT, LOG etc (which are described in chapter 5 of GPD-1). For SAGEM, these are CPU, CMF, NSM, NIR, NWE and KZC. These extra options are described below. However you will find that **the default options** (that is the program settings you get if you do not choose any options) are suitable in most standard cases.

Note that, as for any other GEMPACK program (see section 5.4.2 of GPD-1), **online help** is available. For example, to find information about the SAGEM option **NIR**, just respond '**?NIR**' when prompted for your choice of options. Responding '**??**' will give online help for all options.

14.7.1 SAGEM Options

SAGEM can be run using a Command file. To do this, select option

CMF Input from a GEMPACK Command file

and give the name of the Command file. Further details are given in section 2.12.2 of GPD-1 and in section 18.2 about preparing Command files for SAGEM and their syntax. We recommend that you always run SAGEM from a Command file.

Usually SAGEM carries out iterative refinement of the solutions to improve their accuracy. However you can turn iterative refinement off using option

NIR Don't do solution iterative refinement ("NIR = yes ;" in Command files)

See section 12.1 for more details.

If you only wish to set up the closure for a simulation and then save it on an Environment file (see section 5.2), select

NSM No simulation (closure only) ("simulation = no ;" in Command files)

The Environment file created can then be used in later simulations to specify the closure.

On most machines, it is possible to obtain details of the CPU time (actual machine or Central Processing Unit time). Select option

CPU Report CPU times ("CPU = yes ;" in Command files)

to obtain times for the five stages of a simulation:

1) User specification of the simulation,
2) Creation of the right-hand side matrix, that is, the matrix B in equation (3) of section 10.1,
3) Creation of the Left Hand Side Matrix, that is, the matrix A in equation (3) in section 10.1,
4) LU decomposition of the Left Hand Side Matrix, and
5) Calculation and writing of solutions.

However, on some machines, CPU times are always reported as zero, which means that CPU reporting is not available within GEMPACK on this machine.

Normally warnings are given to indicate how well the equations have been satisfied – see section 12.6.1 for details. Option

NWE Don't warn how well equations solved ("NWE = yes ;" in Command files)

will suppress these warnings. However, you should take these warnings very seriously (see section 12.6.1) so we advise you not to suppress them.

KZC Keep zero coefficients ("KZC = yes ;" in Command files)

Zero coefficients are entries in the Equations Matrix which are calculated to be zero from the current data base but may be non-zero for some other data base. Normally zero coefficients are not saved for SAGEM simulations. However it may help the LU decomposition in some models to save them. See also section 12.5 for a further discussion of zero coefficients and similar options in GEMSIM and TABLO-generated programs.

CHAPTER 15

15. Run-Time Errors

Most of the error messages reported by GEMSIM or TABLO-generated programs will be self-explanatory. Below we give information about situations in which you may need more details in order to rectify the problem.

These are when the simulation fails

- because of a singular matrix,
- because of a divide-by-zero error, or
- because of invalid values in powers or in functions such as LOG, or
- because of some other arithmetic problem such as overflow.

These are discussed in sections 15.1, 15.4, 15.5 and 15.6 respectively. The last three errors above can be thought of as different kinds of arithmetic errors. If such an error occurs in a formula, equation, backsolve or update statement, you will need to find which statement the error occurs in, and will then need to rectify the problem. Advice for doing this can be found in section 15.3.

There are other reasons why a simulation may not run to completion.

- See section 3.4 for checks that the Auxiliary files match the TABLO-generated program.
- See section 4.4 for checks on the set and element data when reading data.
- For details on equations-solving problems see section 12.6.

See also the **Frequently Asked Questions** section of the GEMPACK Web site at address:

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

15.1 Simulation Fails Because of a Singular LHS Matrix

Your simulation may fail because the Left Hand Side Matrix (that is, the matrix A in the equation[172] **A y = b**) is singular (that is, not invertible) at some step. A singular matrix is either an indication

- that the **closure** you are using is not valid (even though it has the required number of exogenous and endogenous variables – see section 5.2.7), or
- of **zeros** in your data base, or
- that at least one linearized equation is a linear combination of other equations present in your model.

The Harwell sparse matrix routines (see section 12.1) distinguish between **structurally singular** and **numerically singular** matrices, and an understanding of the difference may help you sort out the problem.

To see the difference between these, consider the (extremely uninteresting) model with just one linearized EQUATION

```
(all,i,COM)        D(i)*s(i) = t(i) ;
```

where COM is the set (c1, c2) and the two values of COEFFICIENT D are read from the data base. Consider the closure in which both components of variable t are exogenous and both components of variable s are endogenous. In this case the Left Hand Side Matrix A (see section 2.13.2 of GPD-1) will be

```
(  D("c1")      0   )
(     0     D("c2") )
```

Clearly A is invertible if and only if D("c1") and D("c2") are both nonzero.

Suppose now that D("c1") is zero and D("c2") is nonzero. Then the matrix A will be singular. Whether it is reported to be structurally or numerically singular depends on whether you are "keeping zero coefficients" in the Equations file, in the sense explained in section 12.5 above. If you are keeping zero coefficients, A will have two entries (that is, possibly nonzero entries), so that A has the shape

```
( *  0 )
( 0  * )
```

and A will be reported as being **numerically singular**. This means it is singular but

might be nonsingular for different values of the entries (those marked * above).

If, on the other hand, you are not keeping zero coefficients so that A has the shape

```
( 0  0 )
( 0  * )
```

then A will be reported to be **structurally singular**. This means that

there are no values of the entries (marked * above) which could make the matrix invertible.

See also section 15.2 for another definition of structurally singular.

Thus, if you find that the LHS Matrix is structurally singular, and if you are keeping zero coefficients in the current step, this means that the matrix will never be invertible, whatever values you have on the data base, which suggests that your closure is invalid. If, on the other hand, it is numerically singular and you are keeping zero coefficients in the current step, this means that it might become invertible if you changed your data.

This is another reason for us providing options **IZ1** and **KZ2** (see section 12.5 above). If the LHS Matrix is reported as being structurally singular at step 1 and you are using the default option IZ1, you may want to run the program again, this time deselecting option **IZ1** (that is, putting "IZ1 = no ;" in your Command file). If the matrix is then reported as being numerically singular, this indicates that the closure might be valid and may be

[172] At each step of a multi-step simulation, a system of linear equations Ay = b must be solved, as explained in deriving equation (3) in section 2.13.2 of GPD-1. The matrix A and vector b change from step to step as the data is progressively updated, as explained in section 6.2.

failing because of the particular values in your data base. Similarly, it may occasionally help to select option **KZ2** (that is, put "KZ2 = yes ;" in your Command file) if the singular matrix is at step 2 or more.

15.1.1 Trying to Fix a Singular Matrix Problem

The problem of a singular matrix, either structurally singular or numerically singular, can be tricky to analyse and fix. There is no fool-proof general-purpose method for tracking down and eliminating these problems.

However there are several thing you can try.

1. If you are working with a model which has a standard closure or some closure that is non-singular, start from that closure and use swaps to move towards the troublesome closure. Make one swap at a time. The first time you get a singular matrix, examine carefully the last swap to try to identify the problem.
2. Examine the **rank** of the matrix if this is printed out as part of the error message. The rank for a non-singular matrix should be equal to the number of endogenous variables you are solving for. For a singular matrix, the rank is less than the number of endogenous variables (and the difference is called the **rank deficiency**). This gives you some indication of the number of equations which are not independent. Suppose you are dealing with a model containing sets COM and IND of sizes 40 and 38 respectively. If the rank deficiency is 40, then you can try making swaps of vector variables defined over the set COM to try to remedy the problem. On the other hand, if the rank deficiency is 38, maybe a swap of vector variables defined over IND is required. If the rank is only one or two short, you can try swapping macro variables.
3. If the rank is just a few less than the number of endogenous variables, the problem may be caused by just one or two zeros in your data. For example, if you are using the equation
   (all,k,COM) P(k) * y(k) = Q(k)*z(k)
   to solve for the variable y(k), this is only valid if the coefficient P(k) is non-zero **for all values of k in the set COM.** There may be just the occasional value of k where P(k) is zero and this may be causing the singularity of the matrix. You may be able to overcome the problem by changing the value of a data item P(k) from zero to a small nonzero value.[173]
4. Consider switching the values of options IZ1 and KZ2, as discussed in section 15.1 above.
5. If your matrix is numerically singular, you may gain more information by switching to the alternative sparse-solving routines.
   If you are using MA48, try using MA28 by putting the statement "m28=yes ;" in your Command file.
   If you are using MA28, try using MA48 by putting the statement "m28=no ;" in your Command file.
   However if the model is singular with one of these but not with the other, you should be highly sceptical about the validity of the results. You may find that the results are nonsense.
6. If you have different data sets for the same model it may help to try the same closure with a different aggregation. If the model solves with slightly different data, zeros in the data may be the cause of your numerical singularity.
7. If you are building a new model, it may be easier to build it in modules and obtain a closure step by step in the course of development.
8. Equation-solving problems as described in section 12.6 may be caused by a matrix which is very nearly singular.
9. Look at the following section 15.2 for more details on structural singularities.

15.2 Structurally Singular Matrices

In this section we give information which may be relevant if the LHS Matrix is structurally singular.

Recall that the determinant of an NxN matrix is the sum of N! (that is, factorial N) terms, each term being plus or minus a product of N entries from the matrix, one entry from each row and one from each column.

[173] This may mean that the value of y(k) is not really determined correctly for this k value but in the case of a percentage change in an approximately zero variable, this may not affect your simulation results.

For example, the determinant of the 3x3 matrix A with entry a_{ij} in row i and column j is equal to the sum of the 6 terms

$$a_{11}\ a_{22}\ a_{33} \ - \ a_{11}\ a_{23}\ a_{32} \ - \ a_{12}\ a_{21}\ a_{33} \ + \ a_{12}\ a_{23}\ a_{31} \ + \ a_{13}\ a_{21}\ a_{32} \ - \ a_{13}\ a_{22}\ a_{31}$$

The **structural rank** R is defined as the maximum number of nonzero entries a_{ij} , at most one from each row and at most one from each column. The structural rank of the matrix tells you how far the matrix is from being invertible. Any invertible NxN matrix has nonzero determinant and so has structural rank equal to N.

For example, the 3x3 matrix whose nonzero entries are shown by * below has structural rank 2.

[* 0 0]

[* * *]

[* 0 0]

There are several ways of taking 2 nonzero entries from different rows and columns from this matrix. For example, $a_{11}\ a_{22}$ or $a_{11}\ a_{23}$ or $a_{31}\ a_{22}$ or $a_{31}\ a_{23}$ but it is not possible to find 3 such nonzero entries.

When the LHS Matrix is structurally singular, it is not possible to solve the model (with the given closure and data).

The difference between N (the number of endogenous variables) and R (the structural rank) tells how far the LHS Matrix is from being invertible. This difference is sometimes called the **rank deficiency**.

Suppose that the LHS Matrix is structurally singular and that the rank deficiency is equal to D. Then it is possible to identify D variables and D equations with the following properties:[174]

1. If you use the N equations to solve for the remaining N-D variables, it is not possible to solve for any of the identified D variables.
2. The D identified equations add no information to the remaining N-D equations in the sense that, if you used the remaining N-D equations to solve for the remaining N-D variables, the D equations cannot be used to solve for any of the identified D variables.[175]

For example, consider the structurally singular 3x3 matrix above with structural rank 2. Suppose that the equation names are eq1, eq2 and eq3 associated with the 3 rows (in order) and that the variable names are x1, x2 and x3 associated with the 3 columns (in order).[176]

- If you used the first 2 equations to solve for x1 and x2, it is not possible to use the third equation eq3 to solve for the remaining variable x3. Hence variable x3 and equation eq3 are one variable/equation pair which might be identified.
- Equally well, if you used the last 2 equations to solve for x1 and x3, it is not possible to use the first equation eq1 to solve for the remaining variable x2. Hence variable x2 and equation eq1 are another variable/equation pair which might be identified.

Thus, although D variables and equations can be so identified, there is nothing unique or special about the variables and equations so identified. So, although GEMPACK identifies D such variables and equations when the LHS Matrix is structurally singular, we suggest that you do not place too much emphasis on the exact variables and equations.

---

[174] We extract information about the structural rank and the identification of these D variables and equations from the Harwell routines (see section 12.1).

[175] In the language of linear algebra, the D equations are linearly dependent on the remaining N-D equations.

[176] See section 2.13.1 of GPD-1 for the way in which the equations of the model are associated with the rows of the Equations Matrix or the LHS Matrix and for the way in which the endogenous variables of the model are associated with the columns of the LHS Matrix. [The LHS Matrix is the matrix A in section 2.13.2 of GPD-1. It consists of the columns of the Equations Matrix corresponding to the endogenous variables in the given closure.]

15.2.1 Solving Modified Equations when LHS Matrix is Structurally Singular

When GEMSIM or a TABLO-generated program finds a structurally singular LHS Matrix on the first step of a simulation, it reports the structural rank R and the rank deficiency D and identifies (one) set of D variables and equations as described above.

You can ask the software to try to solve a modified system of equations (as described below).[177] We offer this possibility since, once you have a solution, you may be able to use the relevant tools (for example, ViewSOL and/or AnalyseGE) to assist you to find the cause of the singularity. If you request the software to solve the modified equations, you should be aware that the solution produced

is NOT the solution to your original model

and so should be treated with appropriate caution.

To request the software to solve a modified system, you need to add the statement

structurally singular solve modified equations = yes ;

to your Command file. [The default is "no".]

The modified system of equations is produced as follows.

The D variables identified (see above) are given the value zero.

The D equations identified (see above) are modified to ensure that the D identified variables are given the value zero.

The software attempts to solve the modified system which consists of the remaining N-D equations plus the D modified equations.

> Suppose by way of illustration that the D variables identified are the last D variables and that the D equations identified are the last D equations. Then the original LHS Matrix will be of the form
>
> [M1 M2]
>
> [M3 0]
>
> where M1 is RxR, M2 is RxD and M3 is DxR. The 0 block is of size DxD. The entries must be all zero here or else the structural rank would be more than R.
>
> In this case, the modified system of equations solved has LHS Matrix equal to
>
> [M1 M2]
>
> [0 I]
>
> where I denotes the DxD identity matrix (all diagonal entries are 1, all off-diagonal entries are zero).
>
> If the original RHS vector is equal to
>
> [B1]
>
> [B2]
>
> where B1 is of size Rx1 and B2 is of size Dx1, then the modified RHS vector is set equal to
>
> [B1]
>
> [0]
>
> The lower 0 block of size Rx1 ensures that the identified D variables have value zero when the modified system is solved.

The resulting modified system may still be numerically singular (but should not be structurally singular).

A case where the solution to the modified equations may be helpful is where you are checking the homogeneity of a model (see section 13.2 of GPD-4). If the rank deficiency is small (it may be just 1), the identified D variables will only occur in a small number of the equations. Hence the solutions for the other variables in the

---

[177] The reporting of the sets of D variables and D equations, and the ability to solve a modified system were introduced in Release 8.0.

modified system may be very similar to what you would have obtained if you had been able to solve the original equations. Accordingly you may still gain useful information from the solution to the modified equations.

This idea of solving a modified system is one with which we have had only very limited experience. We will welcome correspondence from users who try it and will be particularly interested to hear if the modified solution was helpful or a hindrance.

15.3 Finding Where an Arithmetic Error Occurs and Rectifying It

Here we are discussing

- divide-by-zero errors (see section 15.4),
- errors arising because of invalid values in powers or in functions such as LOG (see section 15.5), and
- other arithmetic problems such as overflow (see section 15.6),

which occur in a formula, equation, backsolve or update statement. When such an error occurs, you will need to find which statement the error occurs in, and will then need to rectify the problem. This section contains advice for doing this.

First you must obtain a Log file from the run. If necessary, rerun the simulation in order to do this (putting "log file = yes ;" in your Command file – see section 2.6). If the error occurs during the first step of a multi-step simulation, you will usually know in which statement the error has occurred since the statements being carried out are echoed one at a time. If the error occurs on a subsequent pass, you may need to add "eaa = yes ;" (see section 6.1.10) to your Command file to get the corresponding information for the relevant pass. Then you can identify the statement in which the error occurred since it will be in **the last statement echoed to the log file**. For example, if your log file ends with

Formula for 'COEF1'
Formula for 'COEF2'
<Error message>

you should expect that the error has occurred while doing the formula for COEF2. If the error occurs during an equation, the log file will help you determine the part of the equation since the different submatrices (see chapter 13 of GPD-4) are echoed to the log file one at a time. [The arithmetic associated with the submatrix for variable 'x' in equation 'eq1' is the arithmetic in the terms involving variable 'x' in that equation.]

Then you can look at this statement in your TABLO file. It may be helpful to write or display (using "xwrite" or "xdisplay" statements – see section 6.6) the values of all Coefficients occurring in the statement where the error occurred. This should help you identify for what commodity or industry etc the error has occurred. You may need to modify the data or theory.

If the error occurs at step 2 or later, you may want to add to your Command file, the statement

```
dws = yes ;                ! see section 6.1.9
```

to show the values of coefficients at all steps of a multi-step simulation.

15.4 Division by Zero Errors

Division by zero is only allowed at certain times, namely when calculating FORMULAs provided the ZERODIVIDE defaults are set appropriately (as described in section 4.13 of GPD-2).

GEMSIM or your TABLO-generated program may stop, reporting that division by zero was attempted when this was not allowed. Suggestions for finding where this occurs (in which formula, equation etc) are given in section 15.3.

Once you have found where the problem occurs, you can probably tell which division caused the problem. You may need to examine the values of the COEFFICIENTs in the denominator (perhaps by adding the relevant XWRITE or XDISPLAY statements – see section 6.6 – in your Command file). This should help work out why the error occurred. Fixing the problem may involve changing the formula etc in question to allow for zeros in the data, or involve adding appropriate ZERODIVIDE statements to the TABLO Input file or it may involve changing the base data and/or closure. In sorting this out, recall that division by zero may be allowed in FORMULAs on your original TABLO Input file, but is never allowed in EQUATIONs, backsolves, UPDATEs or in FORMULAs put in by the system during condensation (see section 2.3.2 of GPD-2) to simplify the subsequent arithmetic.

If you have condensed the model, identifying where the division by zero happens may be somewhat harder than described above since

- the equations, backsolves or updates being calculated may be different from the corresponding expressions in the original TABLO Input file because of substitutions made in them, or
- there may be extra coefficients, formulas and backsolves introduced by TABLO during condensation to simplify the subsequent arithmetic (see section 2.3.2 of GPD-2).

In the first case above, you can see the current (that is, the post-condensation) form of the relevant equation or backsolve when running TABLO by selecting option 's' ("See a summary of the model") after Condensation and before the Code stage; after selecting 's', then select '3' or '4' to see the relevant equation or update. (Backsolves are just equations rewritten.) If the division occurs during a formula introduced by TABLO to simplify the arithmetic, you can see the formula for the relevant coefficient by looking in the Information file (the part that reports condensation actions).

15.5 Error Trapping for Invalid Powers, Logs etc

In versions of GEMPACK prior to Release 7.0, certain invalid arithmetic operations (for example, taking the LOG or square root of a negative number, or raising zero to a negative power) were not trapped by the GEMPACK programs. Rather they were trapped by the operating system. From Release 7.0, GEMPACK traps for these when calculating LOGs, square roots and raising to a power (as in A^B). Such run-time errors will now end with GEMPACK errors, resulting in better diagnostics for users. [For example, the error message may say that you are attempting to take the square root of a negative number.]

These sort or error will nearly always occur while the software is calculating a formula, equation, backsolve or update. See section 15.3 for advice on how to determine where the error occurs and how to rectify it.

15.6 Overflow and Other Arithmetic Errors

There are other arithmetic errors which can occur during a simulation.

The most likely one is **arithmetic overflow**. This occurs when a calculation leads to a number which is larger than can be handled by the program. On Windows PCs, the programs can process real numbers ranging in magnitude up to about $3.4*10^{38}$. On other computers (including Unix machines), there is a limit (usually of a similar size) on the size of real number that can be handled. Overflow occurs when the result of a calculation involving real numbers is larger than this in magnitude. For a simple example, consider the formula

```
Formula   X3 = X1 * X2 ;
```

If X1 is equal to $3.0*10^{30}$ and X2 is equal to $2.0*10^{10}$, then X3 should equal $6.0*10^{40}$. Since the result exceeds the largest real number which can be handled, overflow occurs.

Arithmetic overflow of this kind is a serious error. Once it has occurred, the calculation cannot sensibly proceed so must be stopped with an error message.

In versions of GEMPACK prior to Release 7.0, the GEMPACK programs did not trap for arithmetic overflow errors. Rather they were trapped by the operating system. This meant that the errors were not shown in the log file produced by GEMPACK.

From Release 7.0,

- on Windows PCs, GEMPACK traps for arithmetic overflow. Such run-time errors will now end with GEMPACK errors, which will show on the GEMPACK log file. [For example, the error message may say that arithmetic overflow occurred doing a formula, and the error message will include the name of the Coefficient whose values are being calculated.]
- on other machines (including Unix machines), GEMPACK programs may still not be able to check for arithmetic overflow. Check with your GEMPACK Manager for details.

When GEMPACK programs report arithmetic overflow, they will usually tell you where it occurred. This may be in a formula, equation, backsolve or update; if so, see section 15.3 for advice about confirming where this occurred and for rectifying the problem. Alternatively, overflow may occur while doing some other part of the calculations. Possibilities include carrying out the extrapolation, or updating data during extrapolation. In these cases, the software will often be able to indicate the variable and component number where the error occurred.

Overflow errors usually indicate some problem that you should be able to correct, possibly with your model or its data, or with the simulation (shocks, closure, solution method etc).

15.6.1 Other Sorts of Arithmetic Errors

These are not at all likely. It is possible that a division-by-zero error can occur in a situation (outside formulas etc). It is also theoretically possible to have integer overflow (which means that an integer calculation produces a result larger than the largest integer which can be handled); however this is most unlikely and would indicate a bug rather than something you as a user can correct. GEMPACK programs can trap for some of these errors. If so, the error message will indicate the sort of problem that occurred.

CHAPTER 16

16. Simulations for Models with Complementarities

This chapter contains details about carrying out simulations with models which contain complementarities.

Complementarity statements can be used[178]

- to model various quotas explicitly, including import and export quotas, tariff-rate quotas and output quotas.
- to ensure that investment stays non-negative (see section 16.10).
- to accurately solve models containing piecewise linear functions. Examples include MAX, MIN, ABS and a progressive income tax schedule (see section 16.9).

Several example models containing explicit complementarities, and simulations with them, are provided with GEMPACK. These include versions of Miniature ORANI and GTAP with import volume quotas and/or tariff-rate quotas added. These example models and simulations are described in more detail in sections 16.4 and 16.8 below. An example with ORANIGRD[179] illustrating one way of ensuring that investment stays non-negative can be found in section 16.10. Reference is made to these examples to illustrate various general points made in this chapter.

The syntax and semantics of Complementarity statements in TABLO Input files are discussed in sections 3.19 and 4.14 of GPD-2.

The basic ideas behind the method used in GEMPACK to solve models containing complementarities (see section 16.1) are due to Mark Horridge. We are grateful to Mark for suggesting that we automate these to give a general-purpose algorithm for solving models containing complementarities. The recent paper *A Practical Method for Explicitly Modeling Quotas and Other Complementarities* (Harrison, Horridge, Pearson and Wittwer, 2002) also describes these ideas.

GTAP Technical Paper No. 18 [Elbehri and Pearson (2000)] and GTAP Technical Paper No. 4 [Bach and Pearson (1996)] contain details of previous work on these types of problems.

The definition of a Complementarity (see section 3.19 of GPD-2) is the same as that used elsewhere. For example, see section 5 of Ferris and Kanzow (2002), where they define a Mixed Complementarity Problem. The special case in which there is only a lower bound of zero is generally known as a Nonlinear Complementarity Problem [see, for example, section 1 of Ferris and Kanzow (2002)].

Other algorithms exist for solving problems involving complementarities in economic models, including PATH [see, for example, Dirkse and Ferris (1995)] and MILES (see Rutherford (1993)].

In this chapter we have collected the details you will need to know in order to model complementarities explicitly. The basic ideas are introduced in sections 16.1 and 16.2, some important examples of Complementarity statements are given in section 16.3, and hands-on examples of complementarity simulations are given in section 16.4. Further general information is given in sections 16.5 to 16.7. Several more detailed examples are given in section 16.8 – if you need to model various sorts of quotas, you will probably find an example there which goes at least somewhat in the direction you need to go. Section 16.9 contains information about modelling piecewise linear functions. Section 16.10 describes an extension to ORANIGRD to ensure that investment stays non-negative.

[178] Complementarity statements and the explicit modelling of complementarities were introduced in Release 8.0.

[179] ORANIGRD is a recursive, dynamic version of ORANI-G.

16.1 The Basic Ideas

The basic ideas behind the method used in GEMPACK to solve models containing complementarities are easy to understand.

GEMPACK obtains accurate solutions via extrapolation. This only works when the equations of the model are analytical (that is, differentiable). However the equation specifying a complementarity is not differentiable.[180] This is the basic problem for solving complementarities via GEMPACK.

Mark Horridge's first idea is that if only we knew the post-simulation states (for example, whether the quota is binding or not for each commodity), it would be easy to set up the simulation in such a way that the standard extrapolation methods used in GEMPACK would work. The second idea is that it should be possible to find the post-simulation states via a suitable many-step Euler calculation.

16.1.1 If Only We Knew the Post-Simulation States

Consider an import quota example.

> Suppose that we knew, for each commodity, whether or not the import quota will be binding after the shocks have been applied. For those commodities for which the quota will be binding, the post-simulation import volume will be equal to the quota volume. For those commodities for which the quota will be not binding, the complementarity variable (the extra power of the tariff due to the import quota) should have the value 1. If we could arrange for these two to happen, we could ignore the troublesome equation which expresses the complementarity. But we can arrange for these to happen by modifying the closure and shocks. Take the same closure and shocks as in the original simulation, then modify them as follows.
>
> - For those commodities for which the quota will be binding, make the import volume exogenous (it is probably endogenous in the original simulation) and shock it to the import quota volume.
> - For those commodities for which the import quota will be not binding, make the complementarity variable (the extra power of the tariff due to the import quota) exogenous and shock it to the value 1.
>
> Because the complementarity is the only equation which is not differentiable, and we have arranged for it to hold via the modified closure and shocks, we can ignore that equation when we solve the modified simulation. Thus we can use extrapolation as usual to obtain accurate solutions (since the remaining equations are differentiable).

The same ideas work even if there are several complementarities. The idea is to modify the closure and shocks to guarantee that one of the alternatives allowed by each complementarity holds. Then throw away the complementarity equations, the ones which cause the problems.

16.1.2 Finding the Post-Simulation States

A many-step Euler calculation should be able to find the post-simulation states. Consider again an import quota example.

> In each step, check to see whether the quota is binding or not at the start. If it is not binding at the start, use the part of the complementarity equation which says that the complementarity variable (the extra power of the quota due to the tariff) should stay at its present value (which should be 1) throughout this step. If the quota is binding at the start, use the part of the complementarity equation which says that the import volume should stay equal to the quota throughout this step.
>
> Suppose that the simulation is one in which the import volume for one commodity is rising from below the quota up to the quota. At some stage, the import volume at the end of one of the Euler steps may be higher than the quota. The Euler simulation is set up in such a way that, during the next step, the import

[180] The graph of an equation for a complementarity consists of two or three straight line segments (see, for example, the figures in section 16.2). Because the slopes of these lines are different, the equation is not differentiable at the points where two of these lines meet.

volume is reduced to be exactly equal to the quota (and the extra power of the tariff begins to increase above 1). This correction is introduced via a so-called **Newton correction variable**.

The Figures 16.1.2a to 16.1.2d show this diagrammatically. Consider a shock (shown on the horizontal axis of these figures) which results in an increase in imports.

a. Figure 16.1.2a shows the case in which there is no import volume quota. Then imports increase from the pre-simulation level of I_0 to the post-simulation level of I_1 .

b. Figure 16.1.2b shows the effect of an import volume quota whose value is less than I_1 . Then imports increase until they reach the quota and then level out.

c. Figure 16.1.2c shows the import volumes (with an import quota in place) after each step of a 7-step Euler without a Newton correction term. During step number 4, the import volume jumps above the quota. During the remaining steps, its value stays at the same level (above the quota).

d. Figure 16.1.2d shows the difference that the Newton correction term makes: now imports go back to the quota during step 5 and remain on the quota for the rest of the simulation.

In general, these Newton corrections (bringing the import volume back to the quota volume if it goes above, or bringing the complementarity variable back above its lower bound of 1 if it should go below) ensure that the many-step Euler calculation has a very good chance of correctly finding the post-simulation states.

16.1.3 Combining the Two Ideas

The idea is to first carry out an Euler calculation (as in section 16.1.2) to try to find the post-simulation states. Then modify the original simulation as in section 16.1.1 to carry out an accurate simulation to obtain accurate results using extrapolation.

With the implementation described here, these two steps are automated. You start the simulation as usual by using a Command file. The following two steps are done within the same simulation.

- The software first carries out the Euler calculation. We call this the **approximate run**.
- Then, without any intervention by you, the software automatically modifies the closure and shocks to carry out the **accurate run**. The simulation results are those from the accurate run.

The software carries out checks to ensure that the post-simulation states (as estimated at the end of the approximate run) are those at the end of the accurate run. If not an error message is given. If you use automatic accuracy, the software automatically repeats the subinterval if such a problem occurs during any subinterval.

The rest of this chapter is an elaboration of these ideas.

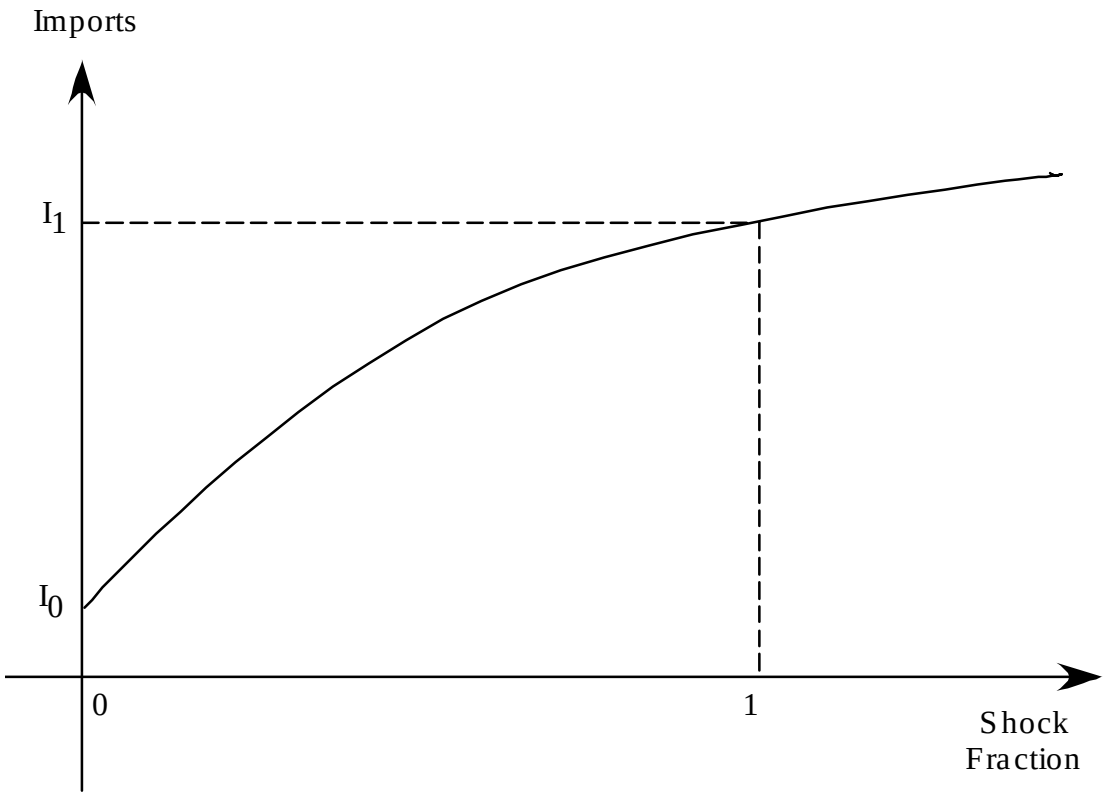


Figure 16.1.2a : No Import Quota

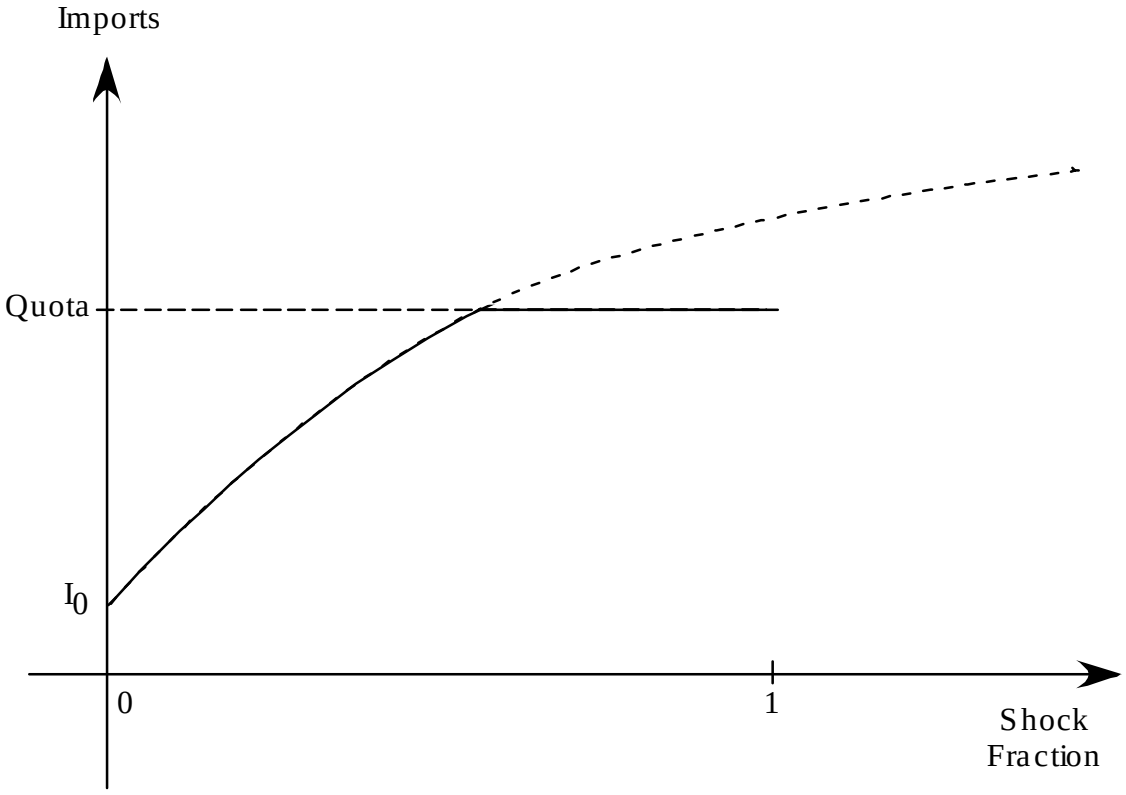


Figure 16.1.2b : Import Quota

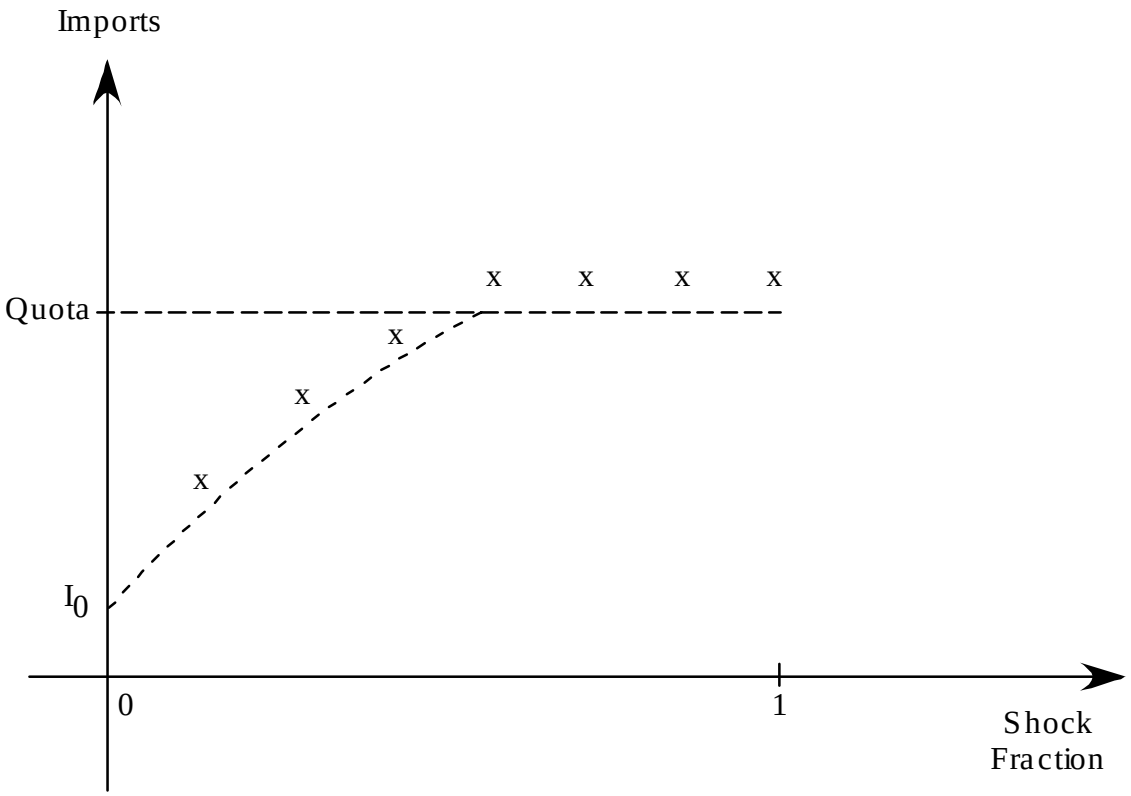


Figure 16.1.2c : Quota – Euler with no Newton Correction

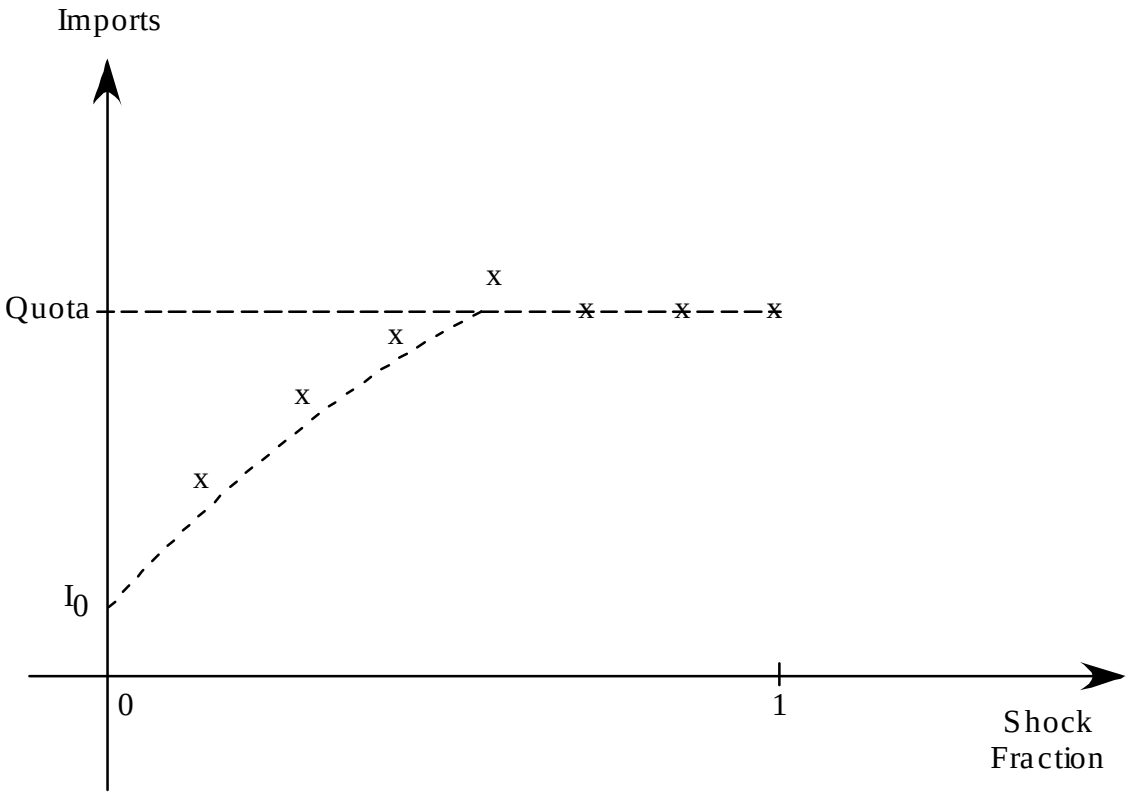


Figure 16.1.2d : Quota – Euler with Newton Correction

16.2 States of the Complementarity

Consider a general complementarity (see section 3.19 of GPD-2)

L <=X<=U ⊥ EXP

which is notation for:	Either	**X = L**	and	**EXP > 0**
	or	**L<=X<=U**	and	**EXP = 0**
	or	**X=U**	and	**EXP < 0** .

See Figure 16.2a for this general complementarity.

This means that there are three states associated with the general complementarity (see Figure 16.2b).

State 1 (Variable Equal to Lower Bound)
The complementarity variable is equal to (or nearly equal to) its lower bound. [The complementarity expression should be positive or zero in this state.]

X = L and **EXP>0**

State 2 (Expression Equal to Zero)
The complementarity expression is zero (or very nearly zero). [The complementarity variable should be in the range bounded by the lower and upper bounds.]

L<=X<=U and **EXP=0**

State 3 (Variable Equal to Upper Bound)
The complementarity variable is equal to (or nearly equal to) its upper bound. [The complementarity variable should be negative or zero in this state.]

X=U and **EXP < 0**

As usual with GEMPACK, your pre-simulation data must be a solution of the model. This means that each component of the complementarity variable and the associated complementarity expression must be clearly in one of the above states in the pre-simulation data.

16.2.1 Complementarities with Only One Bound

First consider a Complementarity with only a lower bound.

```
COMPLEMENTARITY (VARIABLE = X, LOWER_BOUND = L)  comp1  Expression ;
```

This means that:

Either X = L and Expression > 0 **[State 1 – Variable Equal to Lower Bound]**

or X >= L and Expression = 0 **[State 2 – Expression Equal to Zero]**.

[Note that, when there is only a lower bound, there is no state 3, just states 1 and 2.]

Secondly consider a Complementarity with only an upper bound.

```
COMPLEMENTARITY (VARIABLE = X, UPPER_BOUND = U)  comp1  Expression ;
```

This means that:

Either X <= U and Expression = 0 **[State 2 – Expression Equal to Zero]**

or X = U and Expression < 0 **[State 3 – Variable Equal to Upper Bound]**.

[Note that, when there is only an upper bound, there is no state 1, just states 2 and 3.]

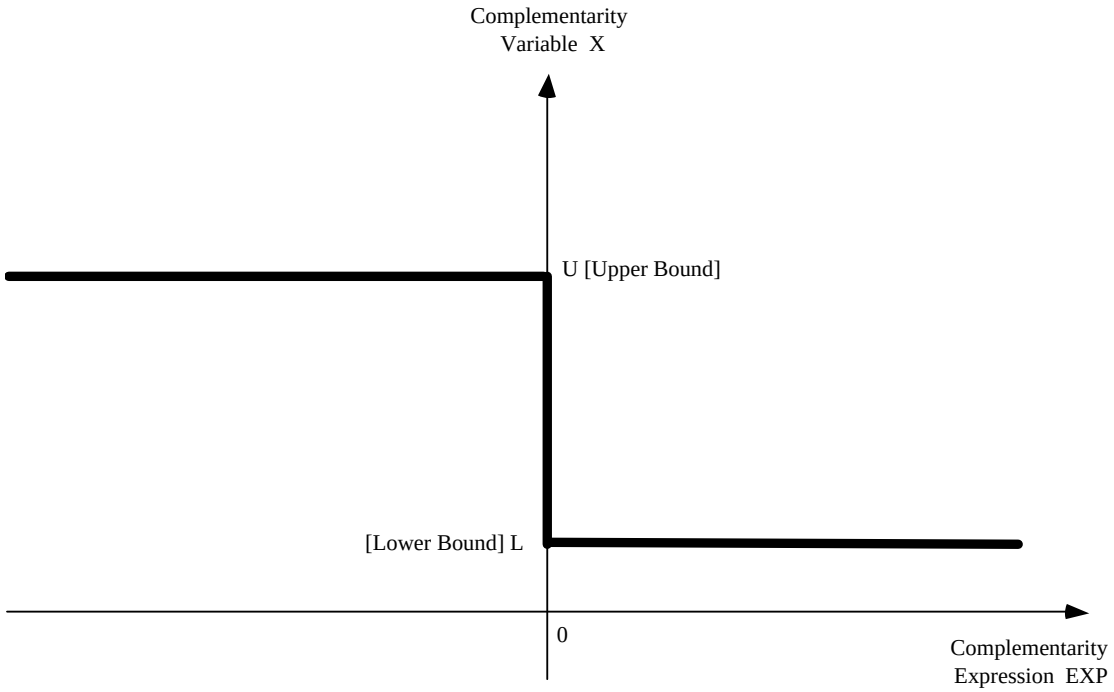


Figure 16.2a : General Complementarity

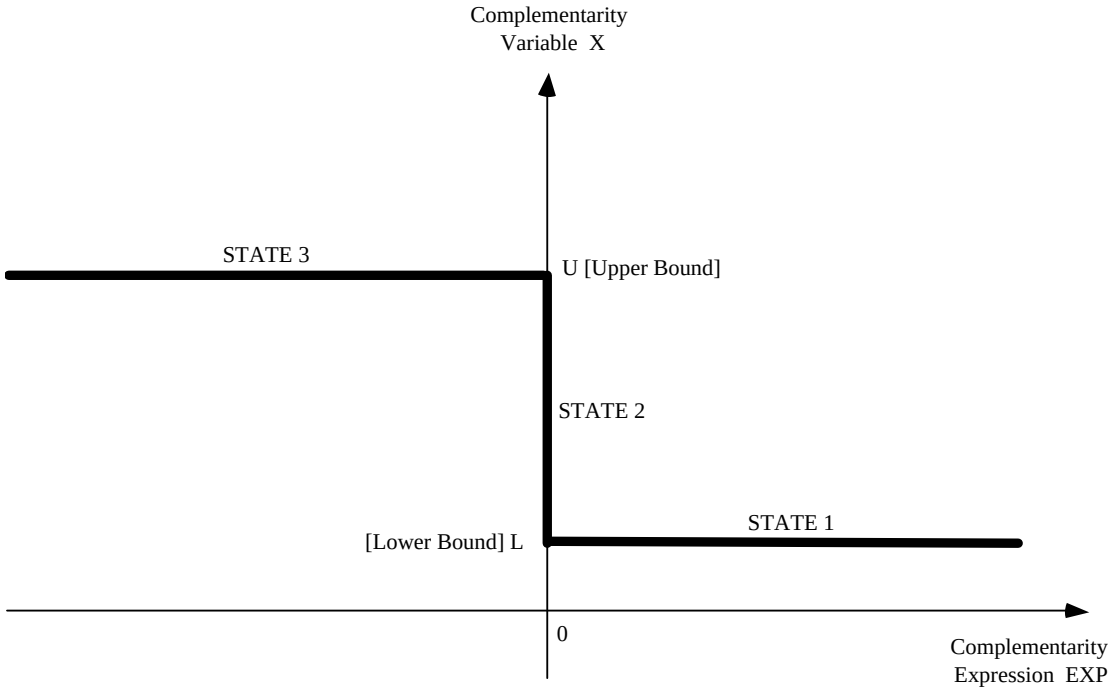


Figure 16.2b : Three States of the General Complementarity

16.3 Writing the TABLO Code for Complementarities

The first step in using complementarities to model import, export or tariff rate quotas or other inequality constraints is to write in your TABLO Input file the additional TABLO code expressing the complementarity. The syntax for Complementarity statements is specified in section 3.19 of GPD-2 and finer details of the semantics are in section 4.14 of GPD-2. In the following section 16.3.1, there is a simple and fairly complete example of import quotas expressed as a complementarity. In section 16.3.2 this complementarity is added to a simple existing model (Miniature ORANI).

16.3.1 Import Quota Example

Volume quotas are applied to imports of commodities, which causes an extra import tariff, due to the quota, to be added to the existing import tariff.

XIMP_QUOTA(i) is the volume import quota and XIMP(i) is the volume of imports.

TIMP_QUOTA(i) is the extra power of the import tariff due to the import volume quota.

To write this complementarity, the variable XIMP_RATIO(i) is introduced, where

XIMP_RATIO(i) = XIMP(i)/XIMP_QUOTA(i)

is the ratio between the current volume of imports XIMP(i) and the quota volume XIMP_QUOTA(i). It is easy to see when the quota has been reached since then XIMP_RATIO is 1.0.

The complementarity expression can be written as 1–XIMP_RATIO(i).

The TABLO notation for the complementarity is :

```
COMPLEMENTARITY
(Variable = TIMP_QUOTA, Lower_Bound = 1)  IMPQUOTA
(all,i,COM)   1.0 – XIMP_RATIO(i) ;
```

If the quota is not binding, XIMP(i) < XIMP_QUOTA(i) so that

XIMP_RATIO(i) < 1.0 that is, 1.0 – XIMP_RATIO(i) > 0

or, if the quota is binding, XIMP(i) = XIMP_QUOTA(i) so that

XIMP_RATIO(i) = 1.0 that is, 1.0 – XIMP_RATIO(i) = 0 .

In the pre-simulation data, the values TIMP_QUOTA(i) and XIMP_RATIO(i) must be in either State 1 or State 2 where the states are defined as:

State 1 (Quota not binding)

TIMP_QUOTA(i) = 1 and 1.0 – XIMP_RATIO(i) > 0

State 2 (Quota binding)

TIMP_QUOTA(i) >= 1 and 1.0 – XIMP_RATIO(i) = 0

It would be wrong to have TIMP_QUOTA(i) = 1 and XIMP_RATIO(i) > 1.0 for any commodity i in the pre-simulation data. (In economic terms this would mean imports XIMP were above quota in the pre-simulation data.)

16.3.2 Example: MOIQ (Miniature ORANI with Import Quotas)

A simple example of import quotas is adding import quotas to the standard Miniature ORANI model MO.TAB.

The various files referred to here can be found in the file **MOQ.ZIP** which should be in the EXAMPLES subdirectory supplied with the GEMPACK software.

The files needed are

- MOIQ.TAB the TABLO Input file,
- MO.DAT the usual MO data file,
- MOIQ_RAT.DAT, the supplementary GEMPACK text data file containing import quota data,
- MOIQ1C.CMF, MOIQ1D.CMF and MOIQ1CX.CMF – Command files for this model.

MOIQ.TAB is the TABLO Input file for Miniature ORANI with import quotas added.

The supplementary data is in the text file **MOIQ_RAT.DAT**. In this data, imports of both commodities are in-quota. Imports of commodity 1 are 0.8 of the quota while imports of commodity 2 are 0.9 of the quota. The Complementarity statement is the same as in the previous section 16.3.1.

Various simulations with MOIQ are used in section 16.4 below to give you hands-on experience with simulations involving Complementarities. If you are keen to carry out a simulation with MOIQ now, go through the steps in section 16.4.1 below, and then return here to read the rest of this section (16.3).

Below we show you the Complementarity-related parts of MOIQ.TAB.

TABLO Code From MOIQ.TAB

```
SET COM # commodities # (c1,c2) ;
SET SOURCE # source of commodities #  ( domestic, imported ) ;

VARIABLE(DEFAULT=LEVELS) ;

VARIABLE
(all,i,COM)(all,s,SOURCE) PCOM(i,s)  # commodity prices # ;
(all,i,COM) PIMP(i)  # import prices (foreign dollars) # ;
(all,i,COM) XIMP(i)  # import quantities # ;
! Change T(i) to TIMP(i)  (WAS) T(i)   # power of import duty #!
(all,i,COM) TIMP(i)   # power of import duty # ;
PHI      # exchange rate # ;
```

etc .....[All statements above here are in the standard MO.TAB.]

```
!       Addition for import volume quotas                          !
!    Set defaults for Levels variables and equations               !

EQUATION(DEFAULT=LEVELS) ;
VARIABLE(DEFAULT=LEVELS) ;
FORMULA(DEFAULT=INITIAL) ;

COEFFICIENT(DEFAULT=PARAMETER) ;

Variable (GT 0) (All,i,COM) XIMP_QUOTA(i) # import volume quotas # ;

Variable (All,i,COM) XIMP_RATIO(i)
  # ratios of import volume to import volume quota # ;

Variable (All,i,COM) TIMP_QUOTA(i)
  # EXTRA power of import tariff due to import quota # ;

FILE (TEXT) quota_ratios # Text file containing quota ratios # ;

READ TIMP_QUOTA FROM FILE quota_ratios ;
READ XIMP_RATIO FROM FILE quota_ratios ;

Formula (Initial)(All,i,COM) XIMP_QUOTA(i) = XIMP(i)/XIMP_RATIO(i) ;

Equation E_XIMP_RATIO

!   (All,i,COM) XIMP_RATIO(i) = XIMP(i)/XIMP_QUOTA(i) ;!
  (All,i,COM) XIMP_RATIO(i)*XIMP_QUOTA(i) = XIMP(i) ;
! Original version of ZPPROFIMP equation must be modified to add TIMP_QUOTA !

FORMULA &  EQUATION ZPPROFIMP
! Zero pure profits in importing commodity i - DPSV (5.36)  !
!(ORIG) (all,i,COM)  TIMP(i) = PCOM(i,"imported") / (PHI * PIMP(i));!
(all,i,COM)  TIMP(i) = PCOM(i,"imported") / (PHI * PIMP(i) * TIMP_QUOTA(i)) ;

COMPLEMENTARITY
   (Variable = TIMP_QUOTA, Lower_Bound = 1)   IMPQUOTA
   (All,i,COM) 1 - XIMP_RATIO(i) ;
```

16.3.3 Specifying Bounds for the Complementarity Variable

When you write down the COMPLEMENTARITY statement, you specify the bounds for the complementarity variable via the LOWER_BOUND and UPPER_BOUND qualifiers. We recommend that you do not also specify these bounds via GE,LE qualifiers (see, for example, section 6.4) when you declare the complementarity variable.

For example, in the Import Quota Example just above, when you declare the complementarity variable TIMP_QUOTA do not include "(GE 1)". Rather declare it via

Variable (Levels) (All,i,COM) TIMP_QUOTA(i) ;

The Complementarity statement ensures that TIMP_QUOTA(i) >= 1.

16.3.4 Scale the Complementarity Expression if Possible

It may be desirable to write down the complementarity expression in a way which makes it well scaled in the sense that its (levels) values are about the same size as those of the complementarity variable. A tariff example will make this clear.

Consider **MOIQ.TAB** (see sections 16.3.1 and 16.3.2 above). The complementarity variable is TIMP_QUOTA(i), the extra power of the import tariff due to the import quota. It has lower bound 1. The complementarity expression we use in MOIQ.TAB is

1 – XIMP_RATIO(i) [1]

where XIMP_RATIO(i) is equal to XIMP(i)/XIMP_QUOTA(i) which is the ratio between the current import volume XIMP(i) and the quota volume XIMP_QUOTA(i). So the values of this complementarity expression are somewhere between zero and 1. Thus the values of this expression and of the complementarity variable TIMP_QUOTA are roughly of the same magnitude.

Contrast the above form of the complementarity expression with perhaps the more natural

XIMP_QUOTA(i) – XIMP(i) [2]

This has the same property as the expression [1] above, namely that it is either positive or zero and is zero when the complementarity variable is greater than the lower bound of 1. But the units of XIMP_QUOTA and XIMP may be quite different from those of TIMP_QUOTA. For example, the natural values to use for XIMP(i) are the values of imports as held on the pre-simulation data base. If these are held in dollars, they may be very large (perhaps in millions or larger).

The two complementarity expressions [1] and [2] above are equally good theoretically. But they may have different numerical properties. When carrying out complementarity simulations, the software needs to add or subtract the values of the complementarity variable and expression in order to estimate which state each component is. This addition or subtraction is more reliable numerically if the numbers being added or subtracted are about the same size. This is why we suggest that you try to scale the complementarity expression.[181]

16.3.4.1 If Possible Scale Bounds and Expression to be About 1

The programs endeavour to find out if the relevant states are fairly exact at the end of the accurate run (see section 16.7.5). Their reports will be more sensible if you are able to scale both the Complementarity Expression and Variable Bounds to of the same order of magnitude as 1. [For example, values between 0 and 5 for the bounds and expression would be ideal.]

We realise that this may not always be possible.

[181] An unscaled expression may work in many cases. For example, we have experimented with using [2] rather than [1] for some simulations. The unscaled [2] works in the cases we have tried. However we expect that there will be some simulations for which the scaled [1] works better in practice than the unscaled [2].

16.3.5 Levels Variables are Required

COMPLEMENTARITY statements require levels variables, both for the Complementarity Variable and in the Complementarity Expression.

This may appear to cause a problem when the rest of the TABLO Input file contains only linear variables and equations. However it turns out to be relatively easy to write TABLO code linking the levels variables needed for the COMPLEMENTARITY statement to the linear variables in the rest of the TABLO Input file. Below we discuss this issue in detail in two of the example models (see sections 16.8.4.2 and 16.8.7.2 below).

You will need to gain experience in writing this "linking" code if you wish to add a Complementarity to an existing linearized model. We suggest that you start by looking closely at the two examples referred to in the previous paragraph. Note also that the LINEAR_VAR= and LINEAR_NAME= Variable qualifiers (see section 2.2.2 of GPD-2) are useful in this context.

16.3.6 Complementarities are Related to MAX and MIN

Complementarities with a single bound (lower or upper) can be expressed in terms of MIN (minimum) and/or MAX (maximum), as we indicate below.

First consider a Complementarity with only a lower bound.

```
COMPLEMENTARITY (VARIABLE = X, LOWER_BOUND = L) comp1 EXP1 ;
```

This means that:

Either X = L and EXP1 > 0, or X >= L and EXP1 = 0.

Equivalently,

Either X – L = 0 and EXP1 > 0, or X – L >= 0 and EXP1 = 0,

which is the same as:

MIN(X – L, EXP1) = 0.

Secondly consider a Complementarity with only an upper bound.

```
COMPLEMENTARITY (VARIABLE = X, UPPER_BOUND = U) comp1 EXP1 ;
```

This means that:

Either X <= U and EXP1 = 0, or X = U and EXP1 < 0.

Equivalently,

Either X – U <= 0 and EXP1 = 0, or X – U = 0 and EXP1 < 0,

which is the same as:

MAX(X – U, EXP1) = 0.

Finally note that MAX(A,B)=0 is equivalent to MIN(-A,-B)=0, so that any complementarity with a single bound can be expressed in terms of either MAX or MIN.

16.3.7 Equivalent Ways of Expressing Complementarities with a Single Bound

Consider the import quota example from above.

```
COMPLEMENTARITY
(Variable = TIMP_QUOTA, Lower_Bound = 1)  ImpQuota
(all,i,COM)   1.0 – XIMP_RATIO(i) ;
```

or

```
COMPLEMENTARITY
(Variable = TIMP_QUOTA, Lower_Bound = 1)  ImpQuota2
(all,i,COM)   XIMP_QUOTA(i) - XIMP(i) ;
```

It is easy to see that this can be written equivalently as

```
COMPLEMENTARITY
  (Variable = XIMP, Upper_Bound = XIMP_QUOTA)  ImpQuota3
    (all,i,COM)   1 - TIMP_QUOTA(i) ;
```

since this means that

Either XIMP(i) < XIMP_QUOTA(i) and 1 – TIMP_QUOTA(i) = 0 [quota not binding],

or XIMP(i) = XIMP_QUOTA(i) and 1 – TIMP_QUOTA(i) < 0 [quota is binding].

More generally, any Complementarity with a single bound can be rewritten with what is the complementarity variable in the original complementarity becoming part of the complementarity expression in the rewritten complementarity, as we explain below.

1. First consider a Complementarity with only a **lower bound**.

```
COMPLEMENTARITY (VARIABLE = X, LOWER_BOUND = L) comp1 EXP1 ;
```

This means that:

Either X = L and EXP1 > 0, or X >= L and EXP1 = 0.

Introduce a new (levels) variable V and an equation saying that

V = EXP1.

Then V >= 0 (since EXP1 is >= 0). Hence the original complementarity can written as

```
COMPLEMENTARITY (VARIABLE = V, LOWER_BOUND = 0) comp1a X - L ;
```

since this means that

Either V = 0 and X – L > 0, or V >= 0 and X – L = 0.

In this alternative way of writing the complementarity, what was the complementarity variable in the original form has become part of the complementarity expression. The original complementarity has been rewritten as a different complementarity statement plus a new equation (the one linking V and EXP1).

Import Quota Example

Consider again the import quota example from above.

```
COMPLEMENTARITY
  (Variable = TIMP_QUOTA, Lower_Bound = 1)  ImpQuota
    (all,i,COM)   XIMP_QUOTA(i) - XIMP(i) ;
```

Following the ideas above this could be rewritten as the equation

```
(All,i,COM) V(i) = XIMP_QUOTA(i)- XIMP(i)
```

and the complementarity

```
COMPLEMENTARITY
  (Variable = V, Lower_Bound = 0)  ImpQuota4
    (all,i,COM)   TIMP_QUOTA(i) - 1 ;
```

2. Secondly consider a Complementarity with only an **upper bound**.

```
COMPLEMENTARITY (VARIABLE = X, UPPER_BOUND = U) comp2 EXP1 ;
```

This means that:

Either X <= U and EXP1 = 0, or X = U and EXP1 < 0.

Introduce a new (levels) variable W and an equation saying that

W = EXP1.

Then W <= 0 (since EXP1 is <= 0). Hence the original complementarity can written as

```
COMPLEMENTARITY (VARIABLE = W, UPPER_BOUND = 0) comp2a U - X ;
```

since this means that

Either W <= 0 and U – X = 0, or W = 0 and U – X < 0.

16.4 Hands-on Examples with MOIQ

In this section we introduce you to hands-on simulations, taking MOIQ (Miniature ORANI with Import Quotas – see section 16.3.2 above) as the example model. We recommend that you run these complementarity simulations now, and check the outputs and LOG file, since then you will be in a better position to follow the discussion about complementarities in the rest of this chapter.

The various files referred to here can be found in the file **MOQ.ZIP** which should be in the EXAMPLES subdirectory supplied with the GEMPACK software.

The files needed are

- MOIQ.TAB the TABLO Input file (see section 16.3.2),
- MO.DAT the usual MO data file,
- MOIQ_RAT.DAT, the supplementary text data file containing import quota data (see section 16.3.2),
- MOIQ1C.CMF, MOIQ1D.CMF and MOIQ1CX.CMF – Command files for this model.

To prepare for these simulations, run TABLO on MOIQ.TAB in the usual way, either producing the TABLO-generated program executable image MOIQ.EXE or the GEMSIM Auxiliary files MOIQ.GSS and MOIQ.GST.

16.4.1 Decreasing a Quota Volume – Command File MOIQ1C.CMF

In the simulation in Command file MOIQ1C.CMF, the quota volume for commodity 1 is decreased by 30%. Since imports of commodity 1 are originally 0.8 of the quota, you would expect this to make the quota binding for commodity 1.

Run the simulation from the Command file.[182] You will see indeed that imports of commodity 1 end up on-quota. The post-sim value of TIMP_QUOTA("c1") is 1.237 (which is larger than its lower bound of 1).

Look at the results using AnalyseGE. In AnalyseGE[183], open the Solution file MOIQ1C.SL4. Search for the COMPLEMENTARITY statement in the TAB file (in the TABmate window of AnalyseGE) and right-click on it. Select ***Show State and Variable Values***.

	STATE-PRE	STATE-POST	VAR-PRE	VAR-POST	EXP-PRE	EXP-POST	LB-PRE	LB-POST
c1	1	2	1	1.2374	0.2	0	1	1
c2	1	1	1	1	0.1	0.049	1	1

The table[184] shows that the **pre-sim** value of the complementarity variable (VAR-PRE) for commodity 1 is 1.0 and pre-sim value of the complementarity expression (EXP-PRE) for commodity 1 is 0.2. Both values are consistent with commodity 1 being in-quota.

---

[182] If you are working on a Windows PC, you can do this via WinGEM.
If are working at the Unix/Command prompt, you can issue the command "moiq –cmf moiq1c.cmf" or "gemsim –cmf moiq1c.cmf".
In either case, the run will produce the log file moiq1c.log since the statement "log file = yes ;" (see section 5.3.3 of GPD-1) is in the Command file.

[183] A Release 8.0 (or later) version of AnalyseGE is needed since complementarities were introduced for Release 8.0 of GEMPACK.

[184] The header C001 in the MOIQ1C.SLC file produced when running the simulation contains this table. This header is put on the SLC file so that AnalyseGE can access it when you click on the Complementarity statement. [Alternatively, you can use ViewHAR to look at this header on the SLC file.]

The table shows that the **post-sim** value for commodity 1 of the complementarity variable (VAR-POST) is 1.2374 and expression (EXP-POST) is 0.0. Both values are consistent with the quota for commodity 1 being binding after the simulation.

The table also shows that the value of the complementarity expression for commodity 2 has decreased from the pre-simulation value (EXP-PRE) of 0.1 to a post-simulation value (EXP-POST) of 0.0490. Thus imports of commodity 2 increase in volume but remain in-quota.

The first two columns show that commodity 1 pre-simulation (STATE-PRE) is in State 1 (Complementarity variable equal to lower bound) and post-simulation (STATE-POST) is in State 2 (Expression equal to zero).

It is also instructive to look at the updated supplementary data in file MOIQ1CQ.UPD. This shows the post-sim values for the complementarity variables TIMP_QUOTA and for XIMP_RATIO.

Look at the Command file MOIQ1C.CMF used in this run. Most of the statements are ones you are familiar with. In fact the only unusual statement is

```
Complementarity steps_approx_run = 20 ;
```

This tells the program how many Euler steps to use in the approximate run (see section 16.6 below).

Now look at the LOG file MOIQ1C.LOG produced from this run.[185] You will see that

- The program first carries out the approximate run. There are 20 Euler steps.
- At the end of the first step of the approximate run, the program reports the initial states for the complementarity IMPQUOTA. The report says

```
[Complementarity IMPQUOTA. Numbers initially in states
 1 and 2 are 2 and 0 respectively.]
```

- During step 14 of the approximate run, the program reports a state change for commodity "c1". The report is:

```
[Complementarity IMPQUOTA("c1"):
  State change from 1 to 2 during this step.
   Occurs after 0.52873969 part of the step.]
```

  This means that the quota for commodity "c1" has gone from non-binding (state 1 – see section 16.2 above) to binding (state 2).

- After the approximate run, the program says that it is beginning the accurate run. It also reports the states and changes found over the approximate run. This part of the LOG file is:

```
===> Beginning accurate complementarity calculation.
[Processing complementarity IMPQUOTA.]
 [Are 1 changing from state 1 to state 2, and
      0 changing from state 2 to state 1.]
 [Are 1 remaining unchanged in state 1, and
      0 remaining unchanged in state 2.]
 [Total numbers in post-sim states 1 and 2 are
   1 and 1 respectively.]
```

- Then the accurate run begins. This is a Gragg 6,8,10-step calculation, as requested in the Command file.
- At the end of the accurate run, the program checks to see if the states are the same as they were at the end of the approximate run. [If they were not, the program would report an error.] Since they are, the program reports the states:

```
Summary of state changes over whole simulation:
[Processing complementarity IMPQUOTA.]
 [Are 1 changing from state 1 to state 2, and
      0 changing from state 2 to state 1.]
 [Are 1 remaining unchanged in state 1, and
      0 remaining unchanged in state 2.]
 [Total numbers in post-sim states 1 and 2 are
   1 and 1 respectively.]
```

[185] This LOG file is produced since the statement "log file = yes ;" (see section 5.3.3 of GPD-1) is in the Command file.

and then indicates that the simulation has ended without error.

- In fact the program changed the closure for the accurate run to make imports of commodity "c1" exogenous (set to the level of the quota). This is done since the approximate run indicated that the quota would become binding for "c1". Notice that you did not need to intervene to achieve this. Nor is there anything in the Command file to suggest how the program should change the closure on the accurate run. [The program "knows" from the Complementarity statement in the TABLO Input file how to change the closure on the accurate run.]

16.4.2 Decreasing both Quota Volumes – Command file MOIQ1D.CMF

In this simulation, the quota volume for commodity 1 is decreased by 30% as for MOIQ1C above. Also the quota volume for commodity 2 is decreased by 25% which makes it also binding.

You might like to run this and check the points referred to above for the MOIQ1C simulation. In particular,

- check the post-simulation states and the post-simulation values of the Complementarity variables and expressions via AnalyseGE.
- check from the log file MOIQ1D.LOG when the states change during the approximate run.

16.4.3 No Approximate Run – Command file MOIQ1CX.CMF

This is the MOIQ1C simulation (see section 16.4.1 above) except that the statement

```
complementarity do_approx_run = no ;
```

(see section 16.6 below) is added to the Command file. This instructs the program not to carry out the approximate run first. The state change which should occur with these shocks (the quota on commodity c1 should become binding) does not happen. Instead the post-simulation values of the complementarity variable TIMP_QUOTA end up being considerably lower than the specified lower bound of 1 for each commodity. The simulation ends with an error. The program reports that the simulation should be rerun, doing the approximate run first, in order to get the correct results.

We suggest that you run this simulation and look at the log file (start at the end) to confirm the points in the paragraph above.

16.5 Other Features of Complementarity Simulations

In this section we comment on some other features of Complementarity simulations.

16.5.1 Closure and Shocks

All components of the complementarity variable which are relevant to the complementarity **must be endogenous**. This must be a consequence of the closure in the Command file for the simulation.

For example, in the Import Quota example in section 16.3.1 above, all components of the variable TIMP_QUOTA must be endogenous.

If the complementarity only applies to subsets of the sets that the complementarity variable is defined over, the complementarity variable must be endogenous over the correct subsets. For example, consider

```
Variable(levels) (All,i,S1),(all,j,S2) V(i,j) ;
Variable(levels) (All,i,LS1),(all,j,LS2) L(i,j) ;
Coefficient(parameter) (All,i,US1),(all,j,US2) U(i,j) ;
Complementarity (Variable = V,
                 Lower_bound = L, Upper_bound=U)  Comp1
      (all, i, T1)(all, j, T2)    X(i,j) – Y(i,j) ;
```

In this case the complementarity variable V(i,j) must be endogenous for all i in the set T1 and all j in the set T2. [Note that T1 must be equal to or a subset of each of S1, LS1 and US1 (see section 4.14 of GPD-2). Similarly, T2 must be equal to or a subset of each of S2, LS2 and US2.]

When TABLO comes across a COMPLEMENTARITY statement, it automatically adds some levels variables (see section 16.7.2 below). When you specify the closure and shocks, you should not list the extra variables automatically added by TABLO in your closure as either endogenous or exogenous. The simulation program (GEMSIM or the TABLO-generated program) automatically sets these variables to be exogenous or endogenous as required, and gives a shock of 1 to the variable $del_comp.

In the example at the end of section 4.14 of GPD-2, all components of $comp1@D and $del_Comp are exogenous and all components of c_comp1@E are endogenous. However these variables should not appear in the closure in specified in the Command file you use to run the simulation.

More details about these extra variables are given in section 16.7.2 below.

In counting up the number of equations to balance with the number of endogenous variables, the complementarity is equivalent to an equation with the same quantifier list.

For example, in the import volume quota example MOIQ.TAB in section 16.3.2,

* three new (levels) variables, XIMP_QUOTA, XIMP_RATIO and TIMP_QUOTA were added.
* one new equation E_XIMP_RATIO was added, and
* one COMPLEMENTARITY was added.

[These are the additions made to the standard MO.TAB file.]

This is like adding 3 COM variables and 2 COM equations. Thus it is necessary to specify one extra COM variable as exogenous when running a simulation.

In the Command file MOIQ1C.CMF (see section 16.4.1), the variable XIMP_QUOTA was added to the exogenous list.

16.5.2 Running TABLO and Simulations

After writing the TABLO code for the model containing a complementarity, use TABLO to implement the model in the normal way. You can carry out condensation if your model is usually condensed – details of restrictions on condensation are in section 4.14.1 of GPD-2.

To run the simulation, just use a Command file in the usual way. The program first does an approximate version of the simulation. Then an accurate version is run, **all as part of the same simulation**. You do not need to specify anything about the approximate run, or carry out any separate steps to make the approximate run occur.

16.5.3 Program Reports State Changes

While carrying out the multi-step Euler approximate simulation, the program reports

- the initial states of the complementarities before the simulation,
- state changes as they occur during the calculation,
- a summary of the state changes that have occurred over the whole approximate run.

You have already seen these reports for the MOIQ1C simulation in section 16.4.1 above.

16.5.4 Program Checks States and Bounds At End of Run

After the accurate run, the program checks

- that the post-simulation states are the same as those found after the approximate run, and
- that the post-simulation values of the complementarity variables are within the specified bounds.

If either of these fails, a fatal error is generated [unless you are doing automatic accuracy, in which case the subinterval is redone, or unless you have the statement "complementarity state/bound_error = warn ;" (see section 16.6) in your Command file].

16.5.5 Checking the Simulation Results

As with any simulation, you must check that the simulation has accurate results, that is that the simulation has converged satisfactorily. Look either at the Extrapolation Accuracy Summaries (see section 7.2) on the Log file or, in detail, at the Extrapolation file (.XAC). [In checking these results, you can ignore non-convergence of the dummy variables added by TABLO (see section 16.7.2 below), since these play no role in the accurate simulation.]

As usual, if the simulation is highly non-linear, it may need more steps (or automatic accuracy) in order to produce sufficiently accurate results. If you think this is why the accurate simulation has not produced good results, increase the number of steps in your Command file, or ask for automatic accuracy. [See section 7.1 for general suggestions. See also section 16.7.4.]

16.5.6 Omitting the Accurate Run

It is possible to do just the approximate run and then stop if you include the statement

```
complementarity do_acc_run = no ;
```

in your Command file. For example, if your model is very large and takes a long time to solve, you may be reluctant to double the total solve time and may be content with the results of the approximate run. Note that, if a Complementarity changes state, because of the way that steps are redone with a Newton correction (see, for example, Figure 16.1.2d above), the approximate run will produce more accurate results than would be produced by a single Euler solution which ignores the complementarity.

There are several restrictions in this case.

- The solution method must be Euler.
- You must ask for a single multi-step Euler solution (any number of steps).
- You must specify only one subinterval.
- You must not specify automatic accuracy.
- Subtotals (see chapter 11) are only allowed if there are no state changes. [If a state change is encountered, a fatal error occurs. This is because the software is not able, at present, to produce correct results for the subtotals when a step is redone.]

16.6 Command File Statements for Complementarity Simulations

1. **To specify the number of Euler steps in the approximate run**:
   The following statement is optional. You only need to add it if the default is not satisfactory. If the following statement is not present the default is the sum of the numbers of steps used in the accurate simulation. For example, if, in the Command file, you ask for Gragg with steps (8, 10, 12), then the default is to use 30 (=8+10+12) Euler steps in the approximate simulation.

   ```
   complementarity steps_approx_run = ddd ;
   ```

   where ddd is the number of Euler steps to use.

   For example, if you want 10 Euler steps in the approximate run, include the statement

   ```
   complementarity steps_approx_run = 10 ;
   ```

   The actual number of steps may be more than you specify. This is because a step may be redone (see section 16.7.3).

2. **Omit the Approximate Run**
   If you think that the shocks will not change any states, you can save time by not doing the approximate run. In this case, you can add the statement

   ```
   complementarity do_approx_run = no ;     ! default is YES
   ```

   to your Command file. The software checks to see if the pre-simulation and post-simulation states are the same, and also to see if the complementarity variables are within the bounds specified in the complementarity statements. If the pre-sim and post-sim states are not the same, or if any complementarity variable is outside the bounds specified, the program reports this and sets a fatal error. Even if there is an error, all files (including the Solution, updated data and SLC files) are kept. Whether there is an error or not, the complementarity arrays are added to the SLC file so that you can look at them in AnalyseGE.

   You must be very careful to check that the results obtained are sufficiently accurate (that is, have converged). [They may not converge if in fact, one or more state changes should occur as a consequence of the shocks.]

   An example is the Command file MOIQ1CX.CMF supplied with MOIQ.TAB (see section 16.4.3 above). With the shocks in this Command file, the complementarity IMPQUOTA("c1") should change from state 1 (in-quota) to state 2 (quota is binding). But, because of the statement "complementarity do_approx_run = no ;" in MOIQ1CX.CMF, this state change is not picked up. Instead the post-simulation values of the complementarity variables TIMP_QUOTA("c1") and TIMP_QUOTA("c2") are reported to be significantly less than the specified lower bound of 1. This is what indicates that a state change should occur. The run ends with a message saying that the simulation has failed and suggesting that you remove the statement "complementarity do_approx_run = no ;" and then rerun the simulation.

3. **Do Not Redo a Step if State Changes During Approximate Run**
   Normally, if one or more states change during a step in the approximate run, the step is redone with a shorter step length so that the state change occurs just before the end of the step (see section 16.7.3). While redoing steps can slow down the approximate run, the extra accuracy achieved is usually worth the extra time. If you do not want steps to be redone in this way, you can include the statement (which we do not recommend)

   ```
   complementarity redo_steps = no ;     ! default is YES
   ```

   in your Command file.

4. **Step Length when Redo Step**
   Normally, if one or more states change during a step in the approximate run, the step is redone with a shorter step length so that the state change occurs just before the end of the step (see section 16.7.3). If a state change occurs very early in the step, the step length of the redone step may be very small. You can control the minimum step length for redone steps by putting a statement of the form

   ```
   complementarity redo_step_min_fraction = <number> ;     ! default is 0.005
   ```

   For example, if you put the <number> equal to 0.01, the step length of the redone step will never be less then 0.01 times the length of the original step.[186] If you do not include such a statement, the program will use the default value of 0.005.

5. **Treat State or Bound Errors as Warnings Only**
   Normally state or bound errors discovered after the end of the accurate run (see section 16.5.4) are regarded as fatal errors (unless you are using automatic accuracy, in which case the subinterval is redone). If you wish these errors to be treated only as warnings, you can include the statement

[186] The length of other steps (those not being redone) is 1 divided by the number of Euler steps in the approximate run. The whole simulation is thought of as having length 1. If there are several subintervals, the length of each step is affected accordingly. For example, if you specify 10 subintervals, each with 5 Euler steps, then each normal step (at least during the first subinterval) will have length 0.02 (one-fiftieth of the whole simulation).

```
complementarity state/bound_error = warn ;        ! default is FATAL
```

in your Command file. If you include this statement (which we do not recommend), it is your responsibility to check your log file carefully for the relevant warnings.

6. **Omit the Accurate Run**
   Normally the approximate run is followed by the accurate run. If you wish to just carry out the approximate run and then stop, you can include the statement

```
complementarity do_acc_run = no ;     ! default is YES
```

in your Command file. See section 16.5.6 for more details about this option. In particular, there are some restrictions on when this is allowed.

16.7 Technical Details about Complementarity Simulations

In this section we give technical details about complementarity simulations. You can probably ignore most of these on a first reading.

16.7.1 How the Closure and Shocks Change for the Accurate Simulation

After running the approximate simulation, the program has estimates of the post-simulation states for each component relevant to each complementarity.

For each component of each complementarity, the program changes closure and gives extra shocks as described below.

1. Suppose that the post-sim state is state 2.

Then the post-sim value of the complementarity expression is zero. This component of the complementarity expression is made exogenous and shocked to the value zero. Since the program can calculate the pre-simulation levels value of this expression, it can calculate the shock.

2. Suppose that the post-sim state is 1.

Then the post-sim value of the complementarity variable must equal the post-sim value of the lower bound.

(a) Suppose that the lower bound is a constant, a parameter or a levels variable for which this component is exogenous. Then the post-sim levels value of the complementarity variable is known (exactly). Accordingly this component of the complementarity variable is made exogenous and shocked to the desired value. Since the program can calculate the pre-simulation levels value of this variable, it can calculate the shock.

(b) Suppose that the lower bound is a levels variable for which this component is endogenous. Then we cannot be sure of the exact post-sim levels value of this lower bound. Accordingly it is not a good idea to proceed as in (a). Instead the variable which is the difference between the complementarity variable and the lower bound is used. [Whenever the lower bound is a levels variable, this variable is introduced automatically by TABLO – see section 16.7.2 below.] The post-sim value of this variable must be zero. This component of this variable is made exogenous and shocked to zero. Since the program can calculate the pre-sim levels value of the complementarity variable and of the lower bound, it can calculate the pre-sim levels value of this variable (the difference between them). Hence the shock can be calculated exactly.

3. Suppose that the post-sim state is 3.

Then, we proceed as in 2. above (just replacing lower bound by upper bound). Either

(a) this component of the complementarity variable is made exogenous and shocked to the known post-sim upper bound, or

(b) this component of the variable equal to the difference between the complementarity variable and the upper bound variable is made exogenous and shocked to the post-sim value zero.

Hence, for every component of the complementarity, one component of one of the relevant variables is made exogenous. [This is either the complementarity expression when the post-sim state is 2, the complementarity variable or the variable which is the difference between this and the lower/upper bound when the post-sim state is 1 or 3.]

To keep the closure correct, ALL components of the complementarity dummy variable (see section 16.7.2 below) are made endogenous. This turns off the non-differentiable equation used as a translation of the complementarity during the approximate calculation.

Since these equations are the only non-differentiable equations in the system, the accurate simulation should converge using extrapolation.

When you carry out a simulation involving one or more complementarities via GEMPACK, the program (GEMSIM or a TABLO-generated program) sets up the new closure and extra shocks after the approximate run and uses this new closure and extra shocks for the accurate calculation.

16.7.1.1 Examples

Simulation in MOIQ1C.CMF

[This simulation was described in section 16.4.1 above.]

For commodity 1 the post-sim state is 2 (while the pre-sim state is 1). This means that commodity 1 of the complementarity expression is set exogenous in the accurate simulation. [The levels, change variable IMPQUOTA@E is introduced by TABLO and set equal to the complementarity expression via the equation E_IMPQUOTA@E – see section 16.7.2 where we describe the extra variables introduced.] Component 1 of this variable IMPQUOTA@E is shocked from its pre-sim value of 0.2 to the post-sim value of zero, so the shock is –0.2.

The post-sim state for commodity 2 is 1. Thus commodity 2 of the complementarity variable TIMP_QUOTA is made exogenous in the accurate simulation. This is shocked from its pre-simulation value of 1.0 (the lower bound) to its post-sim value, also 1.0. Thus the shock for this component is zero.

Simulation in MOTQ1C.CMF

[This simulation is described in section 16.8.2.3 below.]

For commodity 1 the post-sim state is 2 (while the pre-sim state is 1). This means that commodity 1 of the complementarity expression is set exogenous in the accurate simulation. [The levels, change variable TRQ@E is introduced by TABLO and set equal to the complementarity expression via the equation E_TRQ@E – see section 16.7.2 where we describe the extra variables introduced.] Component 1 of this variable TRQ@E is shocked from its pre-sim value of 0.25 to the post-sim value of zero, so the shock is –0.25.

The post-sim state for commodity 2 is 1. Thus commodity 2 of the complementarity variable TIMP_TRQ is made exogenous in the accurate simulation. This is shocked from its pre-simulation value of 1.0 (the lower bound) to its post-sim value, also 1.0. Thus the shock for this component is zero.

Simulation in MOTQ1D.CMF

[This simulation is described in section 16.8.2.3 below.]

For commodity 1 the post-sim state is 3 (while the pre-sim state is 1). Thus commodity 1 of the complementarity variable TIMP_TRQ is made exogenous in the accurate simulation. This is shocked from its pre-simulation value of 1.0 (the lower bound) to its post-sim value of 4.0 (the upper bound). Thus the shock for this component of TIMP_TRQ is 300% (since this is a percentage-change variable).

For commodity 2 the post-sim state is 2 (while the pre-sim state is 1). This means that commodity 2 of the complementarity expression is set exogenous in the accurate simulation. Component 2 of the associated variable TRQ@E is shocked from its pre-sim value of 0.20 to the post-sim value of zero, so the shock is –0.20.

16.7.2 Extra Variables Introduced with Each Complementarity.

Extra variables are introduced for each complementarity as described below. However in general, these extra variables are of no concern to the user. The only restriction from the user's point of view is that all components of the complementarity variable must be endogenous.

(i) **Complementarity expression**. This levels, change variable is called
<comp-name>@E
The levels equation E_<comp-name>@E sets this variable equal to the complementarity expression.

(ii) **Complementarity dummy variable**. This linear, change variable is called
$<comp-name>@D .
This variable appears in the complicated linear equation (see, for example, the linear equation containing $del_Comp shown near the end of section 4.14 of GPD-2) which captures the complementarity statement. This equation is used during the approximate run. This equation is turned off during the accurate simulation (by setting all components of the complementarity dummy variable to be endogenous).

(iii) The **Newton-correction variable $del_Comp**. This is a scalar linear variable. Only one such variable is introduced even if there are several Complementarity statements in the TAB file. This same variable occurs in the linearized equations capturing each complementarity.[187]

(iv) If there is a lower bound and if this lower bound is a levels variable, a levels, change variable is introduced. This is set equal to the difference between the complementarity variable and this lower bound variable. This levels variable is called
<comp-name>@L .
The levels equation E_<comp-name>@L sets this variable equal to the complementarity variable minus the lower bound variable. [This variable will be needed in the accurate simulation if the post-sim state for some component is 1 and if this component of the lower bound variable is endogenous – see case 2(b) in section 16.7.1.]

(v) If there is an upper bound and if this upper bound is a levels variable, a levels, change variable is introduced. This is set equal to the difference between the complementarity variable and this upper bound variable. This levels variable is called
<comp-name>@U .
The levels equation E_<comp-name>@U sets this variable equal to the complementarity variable minus the upper bound variable. [This variable will be needed in the accurate simulation if the post-sim state for some component is 1 and if this component of the lower bound variable is endogenous – see case 3(b) in section 16.7.1.]

During the approximate simulation,[188]
(a) **all components of the complementarity variable must be endogenous** (see section 16.5.1);
(b) the software automatically sets all components of the variable <comp-name>@E (set equal to the complementarity expression) endogenous;
(c) the software automatically sets all components of the complementarity dummy variable <comp-name>@D exogenous and not shocked;
(d) the software automatically sets the variable $del_Comp exogenous with a shock equal to 1;[189]
(e) the software automatically sets all components of the complementarity variables <comp-name>@L and <comp-name>@U (if either exists) endogenous.

---

[187] This variable is used during the approximate run to move variables which go out of their desired bounds back onto the relevant bound. [For example, if the import volume exceeds the quota volume (see, for example, Figure 16.1.2d), this variable is used to bring imports back onto the quota volume during the next step of the approximate run.] This variable plays no role during the accurate simulation. This variable is a NO_SPLIT variable. [The Variable qualifier "NO_SPLIT" is an undocumented alternative to "CHANGE" and "PERCENTAGE_CHANGE" – see section 3.4 of GPD-2. "NO_SPLIT" means that any shock given to this variable is given in full in each step of a multi-step calculation.]

[188] When we talk about variables which are really levels variables being exogenous or endogenous, we should strictly be talking about the associated linear variable.

[189] In some rare circumstances, you may not want this to happen automatically. One such circumstance is when you are carrying out the Base Rerun or the Policy under RunDynam, RunMONASH etc (see section 2.7 of GPD-4). You can use the statement
auto_shocks = no ;
in your Command file to prevent the software adding a shock of 1 to $del_Comp. [Release 8.0 (or later) versions of RunDynam, RunMONASH etc add this statement to the relevant Command files.]

You must not mention any of the variables in (b)-(e) above in your Command file. The closure you set up in the Command file must satisfy (a). [If this does not seem possible, maybe you can express the complementarity in a slightly different way to accomplish this.]

16.7.3 Step May Be Redone During Approximate Run

When a change of state occurs during one Euler step of the approximate run, this step is redone with a shorter step length so that the state change occurs just before the end of the step. This means that the simulation does not overshoot a bound by very much during the approximate run.

For example, in the MOIQ1C.CMF simulation (see section 16.4.1 above), the complementarity IMPQUOTA("c1") changes from state 1 (in-quota) to state 2 (on-quota) after about 0.529 of the 14th Euler step. This step is redone (reducing its length to a little over 52.9% of the original step length). When the step is redone, this complementarity IMPQUOTA("c1") changes state after approximately 0.995 of this new, shorter step. Thus the import volume of commodity "c1" only overshoots the quota by a very small amount on this rerun step.[190]

If two or more state changes occur during one Euler step of the approximate run, the step is redone with a shorter step length so that the state change which occurs first during the step occurs just before the end of the step. While this usually means that only one state change occurs during the redone step, it may happen that two or more occur since they occur at nearly the same stage during the step.

Because some steps may be redone, the actual number of steps in the approximate run may be more than you request in the "complementarity steps_approx_run = … ;" statement (see section 16.6) in your Command file.

16.7.4 Several Subintervals and Automatic Accuracy

As with any other simulation, you can request that the simulation be carried out over several subintervals (see section 7.3) or with automatic accuracy (see section 7.4).

If you specify several subintervals or automatic accuracy,

an approximate run is followed by an accurate run in each subinterval.

[You specify several subintervals by including a statement of the form "subintervals = … ;" in your Command file (see section 7.3).

If you specify automatic accuracy, the software checks the accuracy at the end of each subinterval (as usual). It also checks that the complementarity part of the simulation has worked over this subinterval. For example, it checks that

- the post-simulation states after the accurate run for this subinterval are the same as the post-simulation states after the approximate run for this subinterval.
- the post-simulation values of the complementarity variables are within the required ranges.

If any of these tests fails, the subinterval is redone (with a shorter subinterval length). [An example is 1B5RAAX.CMF which is the same as 1B5RAA.CMF (see section 16.8.4.3 below) except that one subinterval is specified to start with. The attempt to do the whole simulation in one subinterval fails because the post-sim state for (sugar,AFR,USA) is 3 after the accurate simulation but 2 after the approximate simulation. The subinterval length is reduced and the simulation then proceeds ok.]

The advantage of specifying automatic accuracy with a relatively nonlinear simulation (for example, one with relatively large shocks) is that you do not need to experiment to determine how many Euler steps in a single subinterval are needed to satisfactorily predict the post-simulation states. An interesting example in this context is the 1B5R.CMF GTAP simulation (see section 16.8.4.3 below).

---

[190] At the end of the original 14th step, imports of commodity "c1" would have overshot the quota volume by a significant amount. Note also that the rerun step is called the 15th step in the LOG file. After this rerun step, the step length returns to the original length. [In the MOIQ1C case, this corresponds to one-twentieth of the total shock.]

You specify automatic accuracy by including the statement "automatic accuracy = yes ;" in your Command file (see section 7.4). [You can also specify the length of the first subinterval via a statement of the form "subintervals = … ;".]

If the approximate run is having difficulty finding the accurate post-simulation states (perhaps because of large shocks), we strongly recommend that you use several subintervals or automatic accuracy.

16.7.5 Checks that the Pre- and Post-Simulation States are Accurate

At the beginning of each simulation, the software checks that the data has each component of each complementarity accurately in one or other of the possible states. This means checking that the data represents points close to the exact graph of the complementarity. In terms of Figure 16.7.5a, it means that all points lie within some dotted region close to the exact graph of the complementarity.

Similar checks are made at the end of the whole simulation (and at the end of each subinterval if there are several).

If any of the states is not exact (to within some tolerance represented by the dotted lines in Figure 16.7.5a), a warning is issued. At present the software takes no stronger action in these cases and it is your responsibility to check out these warnings.

16.7.5.1 Less Exact States Used Elsewhere by the Software

This subsection contains technical information which you may prefer to ignore. It is included for those who wish to know in a little more detail how the software operates.

During the multi-step Euler approximate calculation, and at the end of the accurate run, the software must decide in which state each component of each complementarity lies. Especially during the approximate run, the points involved may lie some distance away from the exact states as represented in Figure 16.7.5a. The software needs to assign a state for all points in the plane. This is done as indicated in Figure 16.7.5b. All points above the higher diagonal line (with equation V – EXP = U) are assigned to state 3. All points below the lower diagonal line (with equation V – EXP = L) are assigned to state 1. All points between these diagonal lines are assigned to state 2.[191]

[191] Interested readers may like to compare Figure 16.7.5b with Figure 3 in TP18 [Elbehri and Pearson (2000)].

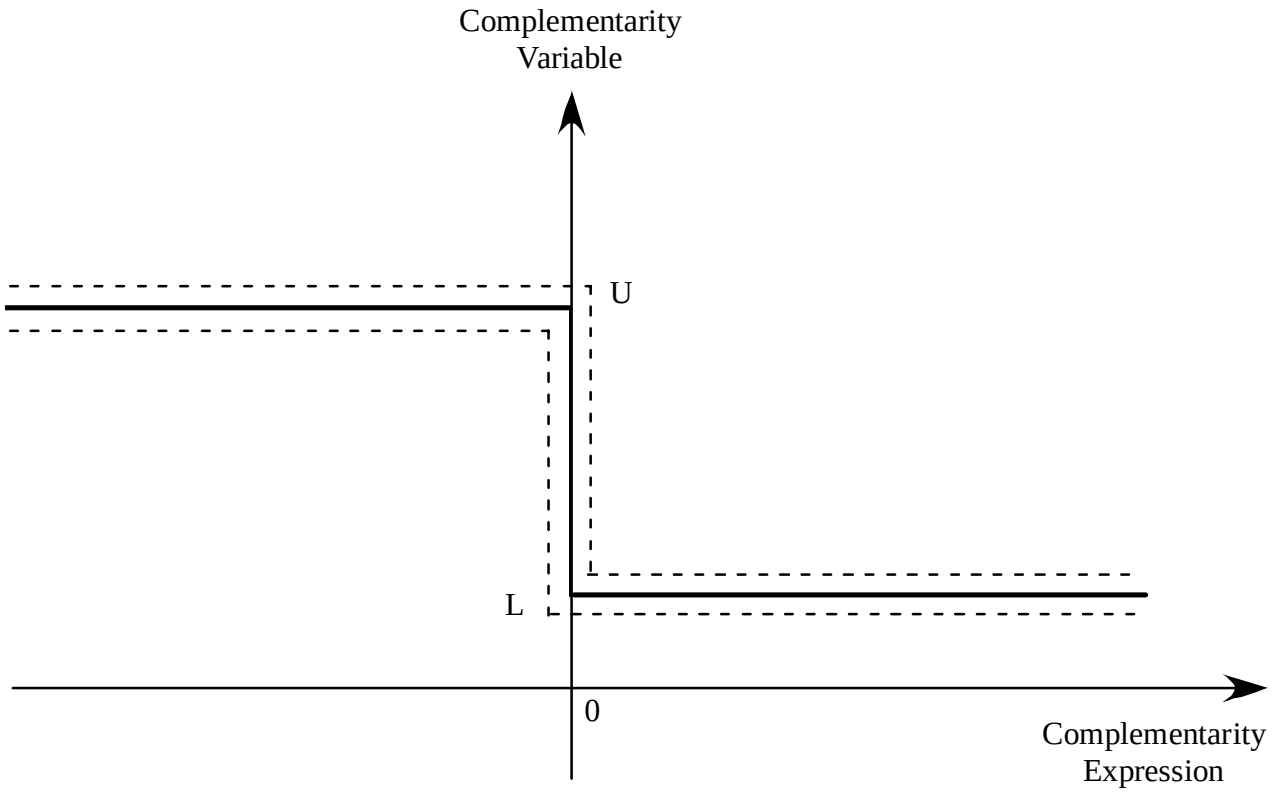


Figure 16.7.5a : Accurate States

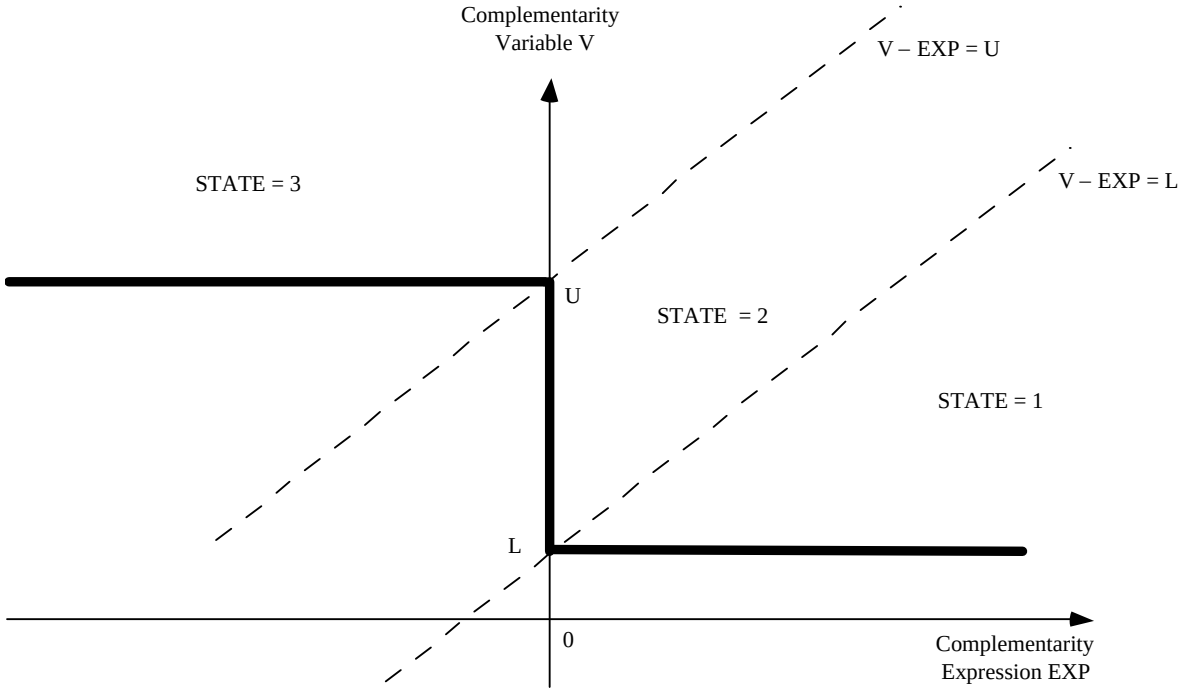


Figure 16.7.5b : Whole Plane Divided into 3 (Inexact) States

16.8 Complementarity Examples

This section introduces several different models containing complementarities. The associated examples files are supplied with GEMPACK in the EXAMPLES subdirectory.

Examples on import quotas and tariff-rate quotas with Miniature ORANI (sections 16.8.1 to 16.8.3 below) are in the zip file **MOQ.ZIP**.

GTAP examples on bilateral tariff-rate quotas and global tariff-rate quotas (sections 16.8.4 to 16.8.6 below) are in **G5TRQ.ZIP.**

G94-QUO.ZIP contains examples from GTAP94 with bilateral import quotas and bilateral export quotas (see sections 16.8.7 and 16.8.8 below).

16.8.1 Example: MOIQ (Miniature ORANI with Import Quotas)

The various files referred to here can be found in the file **MOQ.ZIP** which should be in the EXAMPLES subdirectory supplied with the GEMPACK software.

The files needed are

- MOIQ.TAB the TABLO Input file,
- MO.DAT the usual MO data file,
- MOIQ_RAT.DAT, the supplementary GEMPACK text data file containing import quota data,
- MOIQ1C.CMF, MOIQ1D.CMF, MOIQ1CX.CMF and MOIQ1DR.CMF – Command files for this model.

See sections 16.3.1 and 16.3.2 above for details about the TABLO code and the supplementary data file MOIQ_RAT.DAT, and see section 16.4 above about running the first three of these example simulations and a discussion of results.

The Command file MOIQ1DR.CMF carries out the reversal of the MOIQ1D.CMF simulation.

Also supplied in MOQ.ZIP is TABLO Input file MOI2.TAB. This contains an alternative form of the Complementarity statement to implement an import quota. [The Complementarity statement in MOT2.TAB has an upper bound whereas the Complementarity statement in MOIQ.TAB has a lower bound.] Command files MOI21C.CMF, MOI21D.CMF and MOI21DR.CMF carry out simulations with MOI2.TAB. The results of these are the same as the corresponding simulations based on MOIQ.TAB (as you can check if you carry them out).

16.8.2 Example: MOTRQ (Miniature ORANI with Tariff-Rate Quotas)

The various files referred to here can be found in the file **MOQ.ZIP** which should be in the EXAMPLES subdirectory supplied with the GEMPACK software.

The files needed are:

- MOTRQ.TAB, the TABLO Input file. This is usual Miniature ORANI TABLO Input file MO.TAB with tariff-rate quotas added.
- MO.DAT, the usual data file for MO.
- MOTRQ.DAT, a test data file which contains supplementary data for tariff-rate quotas.
- MOTQ1C.CMF, MOTQ1D.CMF and MOTQ1DR.CMF – the Command files for this model.

Associated with a tariff-rate quota are two import tariff rates. The lower rate (the **in-quota rate**) applies when the import volume is below the quota volume and the higher rate (the **over-quota rate**) applies when the import volume exceeds the quota volume.

> Suppose by way of example, that the *ad valorem* in-quota rate is 25% and that the *ad valorem* over-quota rate is 400%. Then the in-quota power of the tariff is 1.25 and the over-quota power of the tariff is 5.

When the import volume is exactly equal to the quota volume, the tariff rate (the **on-quota rate**) is somewhere between the in-quota rate and the over-quota rate. Import volumes do not exceed the quota volume until importers are prepared to pay the full over-quota rate. As economic conditions become more favourable towards imports, the on-quota rate increases from the in-quota rate towards the over-quota rate. If conditions become extremely favourable to imports, the in-quota rate may rise to be equal to the over-quota rate, and then imports may expand to exceed the quota volume.[192]

In the numerical example above, the *ad valorem* on-quota rate varies between 25% and 400%.

16.8.2.1 MOTRQ.TAB

It is convenient to consider also the extra power of the import tariff due to the tariff-rate quota. In the TAB file MOTRQ.TAB you can see the following levels variables declared.

```
Variable
(all,i,COM) TIMP(i)  # power of import duty #
 ! One plus rate of import duty on commodity 1 – DPSV t(i) ! ;
(all,i,COM) XIMP(i) # import quantities # ;
(GT 0) (all,i,COM) XIMP_TRQ(i)
  # TRQ quota volume # ;
(GT 0) (all,i,COM) TIMP_INQ(i)
  # in-quota power of import tariff # ;
(all,i,COM) TIMP_TRQOVQ(i)
  # extra power of import tariff in over-quota case # ;
(GT 0) (all,i,COM) TIMP_OVQ(i)
  # total power of import tariff in over-quota case # ;
(GT 0) (all,i,COM) TIMP_TRQ(i)
  # actual extra power of import tariff due to TRQ # ;
(GE 0, LE 1) (all,i,COM) XIMP_RATIO(i)
  # Current volume [XIMP] divided by TRQ quota volume [XIMP_TRQ] # ;
```

This makes the distinction between the **total** power of the import tariff in the over-quota case TIMP_OVQ(i) and the **extra** power of the import tariff in the over-quota case TIMP_TRQOVQ(i). These are connected by the equation

```
Formula & Equation E_TIMP_OVQ
  # Total power in over-quota case #
  (All,i,COM) TIMP_OVQ(i) = TIMP_INQ(i) * TIMP_TRQOVQ(i) ;
```

In the numerical example above (where the in-quota power is 1.25 and the over-quota power is 5), TIMP_INQ is 1.25, TIMP_OVQ is 5 and so the extra power TIMP_TRQOVQ is 4 and the equation above says that 5 = 1.25*4.

There is also the distinction between the actual power of the import tariff TIMP(i) and the **extra** power due to the tariff-rate quota TIMP_TRQ(i). These are related by the equation

```
Formula & Equation E_TIMP_TRQ
  # Actual extra power due to TRQ #
  (All,i,COM) TIMP_TRQ(i) = TIMP(i)/TIMP_INQ(i) ;
```

In the numerical example above, the actual power of the tariff TIMP can vary between 1.25 and 5. The extra power of the tariff due to the tariff-rate quota TIMP_TRQ can vary between 1 and 4.

The complementarity statement in MOTRQ.TAB is

[192] An introduction to tariff-rate quotas from a modeller's point of view can be found in GTAP Technical Paper No. 18 [Elbehri and Pearson (2000)].

```
COMPLEMENTARITY
   (Variable = TIMP_TRQ,
    Lower_Bound = 1,
    Upper_Bound = TIMP_TRQOVQ)   TRQ
  (All,i,COM) 1 - XIMP_RATIO(i) ;
```

The complementarity variable is TIMP_TRQ, the extra power of the import tariff due to the tariff-rate quota.

The complementarity expression is 1 – XIMP_RATIO(i), where

XIMP_RATIO(i) = XIMP(i) / XIMP_TRQ(i)

is the ratio between the current volume of imports XIMP(i) and the TRQ quota volume XIMP_TRQ(i).

16.8.2.2 Supplementary TRQ Data in File MOTRQ.DAT

The supplementary data are in the text data file **MOTRQ.DAT**. These data imply that

1. imports of both commodities are in-quota.
2. imports of commodity 1 are 0.75 of the quota volume while
   imports of commodity 2 are 0.8 of the quota volume.
3. the full extra power of the import tariff due to the tariff-rate quota is 4.0 for each commodity. That is, if imports of either commodity exceed the quota volume, this extra power of the tariff will apply.

You might like to examine MOTRQ.DAT in your text editor to see the actual data held there. The relevant levels variables and read statements in the TAB file are shown below.

```
Variable
(GE 0) (all,i,COM) VIMP_TRQ(i)
 # value of quota volume at world prices (foreign currency) # ;
(GE 0) (all,i,COM) VIMPINQ_TRQ(i)
# value of quota volume at in-quota prices (foreign currency) # ;
!-------------------------------------------------------------!
! Extra READs for Tariff-Rate Quotas                          !
!-------------------------------------------------------------!
FILE (TEXT) trq_data # Text file containing TRQ data # ;
READ TIMP_TRQOVQ FROM FILE trq_data ;
READ VIMP_TRQ FROM FILE trq_data ;
READ VIMPINQ_TRQ FROM FILE trq_data ;
```

The values of imports at world prices (as shown in the standard data file MO.DAT for Miniature ORANI) are 9 and 12 for the two commodities. [The total values including duty are 10 and 17, as seen from headers "IIMP" and "IMPH", while duty values are 1 and 5 respectively – see header "DUTY".] The VIMP_TRQ values on MOTRQ.DAT are 12 and 15 respectively. These mean that the corresponding XIMP_RATIO values are 0.75 (=9/12) and 0.8 (=12/15) respectively since

XIMP_RATIO(i) = XIMP(i)/XIMP_TRQ(i) = VIMP(i)/VIMP_TRQ(i)

[where VIMP(i) is the value of imports at world prices] because the common price PIMP(i) cancels out.

Since these XIMP_RATIO values imply that imports are in-quota in each case, the VIMPINQ_TRQ(i) values shown must be 10/0.75 and 17/0.8 respectively, since 10 and 17 are the values including duty.

This explains the values you see on MOTRQ.DAT.

16.8.2.3 Example Simulations with MOTRQ

(i) Decreasing the Quota Volume until imports are on-quota – MOTQ1C.CMF

The simulation in Command file MOTQ1C.CMF decreases the quota volume XIMP_TRQ("c1") for commodity 1 by 30%. After this shock, the quota volume will only be 70% as large as it was initially.

Since initial imports XIMP("c1") are 0.75 of the initial quota volume, after this decrease in the quota volume, you would expect imports of commodity 1 to be on (or over) quota.

If you run the simulation you will see indeed that imports of commodity 1 end up on-quota. The post-sim value of TIMP_TRQ("c1") is 1.116 (which is between its lower bound of 1 and the upper bound of 4).

Run the simulation.[193] Then open the Solution file MOTQ1C.SL4 in AnalyseGE. Right-click on the COMPLEMENTARITY statement and select **Show State and Variable Values**.

	STATE-PRE	STATE-POST	VAR-PRE	VAR-POST	EXP-PRE	EXP-POST	LB-PRE	LB-POST	UB-PRE	UB-POST
c1	1	2	1	1.116	0.25	0	1	1	4	4
c2	1	1	1	1	0.2	0.177	1	1	4	4

The table shows that for commodity 1, the pre-sim value of the complementarity variable (VAR-PRE) is 1.0 and expression (EXP-PRE) is 0.25. Both values are consistent with commodity 1 being in-quota (that is, in state 1 – see the STATE-PRE column in the table). It shows that the post-sim value of the complementarity variable (VAR-POST) and expression (EXP-POST) for commodity 1 are 1.116 and 0.0 respectively. Both values are consistent with commodity 1 being on-quota after the simulation (that is, in state 2 – see the STATE-POST column in the table). This table also shows that the value of the complementarity expression for commodity 2 has decreased from 0.2 (pre-sim) to 0.177 (post-sim). Thus imports of commodity 2 increase in volume but remain in-quota.

It is also instructive to look at the updated supplementary TRQ data in file MOTQ1CQ.UPD. The extra powers of the tariff for over-quota imports do not change – they remain equal to 4.0 for each commodity. But the values of VIMP_TRQ and VIMPINQ_TRQ change for commodity 1. These are the values of the quota volume valued firstly at world prices (VIMP_TRQ) and secondly at the in-quota tariff rate (VIMPINQ_TRQ). Both are held in foreign dollars. For commodity 1 these are reduced by 30% (since the quota volume for commodity 1 has been reduced by 30% and foreign currency prices have not changed). These values for commodity 2 are unchanged since the quota volume and the foreign currency price is unchanged.

If you look at the log file motq1c.log produced in this run, you can see when the state change occurs during the approximate run. [Search for the word **state** in the log file.] You should find that the state for commodity c1 changes from 1 (in-quota) to 2 (on-quota) during Euler step 17 of the approximate run.[194] You can also see a little further down the log file that this step 17 is redone with a shorter length (see section 16.7.3).

---

[193] If you are working on a Windows PC, you can do this via WinGEM.
If are working at the Unix/Command prompt, you can issue the command "motrq –cmf motq1c.cmf" or "gemsim –cmf motq1c.cmf".
In either case, the run will produce the log file motq1c.log since the statement "log file = yes ;" (see section 5.3.3 of GPD-1) is in the Command file.

[194] At this place in the log file you can see

```
%% Test that updated values of XIMP_RATIO LE 1 fails
(Value is 1.0016447)
```

This report occurs because of the range "(GE 0, LE 1)" specified when the levels variable XIMP_RATIO(c) was declared in MOTRQ.TAB. [It is not really necessary to specify the "LE 1" part when declaring XIMP_RATIO since the Complementarity statement should ensure that XIMP_RATIO(c) remains (approximately) in the expected range. In fact, during step 17 of the approximate run, XIMP_RATIO("c1") does overshoot the value of 1. This happens even when the step is redone (see a little further down the log file), when you see that XIMP_RATIO("c1") is equal to 1.0000021.] Since overshooting is expected during the approximate run, the "(LE 1)" qualifier should really not be included when XIMP_RATIO is declared in the TAB file. [We have left it there to let you see the overshooting message above.]

(ii) Decreasing the Quota until the imports are over quota – MOTQ1D.CMF

The simulation in Command file MOTQ1D.CMF decreases the quota volume XIMP_TRQ("c1") for commodity 1 by 70%. This is a larger decrease than in MOTQ1C.CMF.

If you run the simulation you will see that imports of commodity 1 end up over-quota. The post-sim value of TIMP_TRQ("c1") is 4.000 and the post-sim value of the complementarity expression is -0.2678. Both of these values are consistent with commodity 2 being over-quota (since 4.0 is the full extra power of the import tariff which applies to over-quota imports of commodity 1 – see MOTRQ.DAT). [Again you can see these values most easily if you use AnalyseGE.]

The post-sim state of commodity 2 is interesting. You can see from AnalyseGE that this commodity ends up on-quota. Imports of this commodity expand and the post-sim values of the complementarity variable and expression for this commodity are 1.502 and 0.0 respectively.

Again it is instructive to look at the updated supplementary TRQ data in file MOTQ1DQ.UPD.

(iii) Increasing the Quota Volume MOTQ1DR.CMF – Reversing MOTQ1D

In MOTQ1D.CMF the quota volume for commodity 1 was decreased by 70%. The reversal of this in the Command file MOTQIDR.CMF is to increase the quota volume for this commodity by 233.333%.[195]

This simulation starts from the updated data after the MOTQ1D simulation. The shock is a 233.333% increase in the quota volume for commodity 1. You would expect the results of this simulation to be the exact opposite of the results of the original MOTQ1D simulation. And you would expect the post-simulation data after this reversal to be the same as the original MO data and supplementary TRQ data. It is easy to check these expectations, as we explain below.

For the simulation results, consider two examples.

(i) Look at industry activity in the first sector, that is p_Z("c1"). The result in MOTQ1D is a decrease of 13.749372%.[196] The reversal of such a decrease is an increase of 15.9412%.[197] The results from the reversal simulation MOTQ1DR show an increase of 15.9337% which is acceptably close to 15.9412.

(ii) Look at changes in import volumes p_XIMP for the two commodities. The MOTQ1D results show a decrease of 49.280865% in imports of commodity 1 and an increase of 25.0% in imports of commodity 2.[198] The exact reversals of these would be an increase of 97.1642% for commodity 1 and a decrease of 20.0% for commodity 2.[199] The simulation results from MOTQ1DR indicate an increase of 97.1801% for commodity 1 and a decrease of 19.997% for commodity 2. Again these are reasonably close to the exact results. [Of course, you could get even better agreement if you carried out both simulations more accurately – that is, took more Gragg steps.]

---

[195] Suppose the quota volume was 100 originally. [Exactly what value we assume here is irrelevant.] Then, after a 70% decrease the quota volume would be 30. To get back to the original quota volume of 100, the increase must be the opposite of the original decrease, that is an increase of 70 in the quota volume. As a percent, this is (70/30)*100 = 233.333%.

[196] This is the result reported by ViewSOL if you use GEMSIM so carry out the simulation. If you use a TABLO-generated program, the result reported is a decrease of 13.749338 percent. [These small differences are expected since GEMSIM and TABLO-generated programs do their arithmetic in slightly different ways. Indeed, these are the numbers when the simulation is run on a Windows PC. On a Unix machine, they may be slightly different again.]

[197] Suppose the activity level was 100 originally. [Exactly what value we assume here is irrelevant.] Then, after a 13.749372% decrease the activity would be 86.250628. To get back to the original activity of 100, the increase must be the opposite of the original decrease, that is an increase of 13.749372 in the activity level. As a percent, this is [13.749372/86.250628]*100 = 15.94118%.

[198] Again this is the result reported if GEMSIM is used to carry out the simulation, and the number reported is slightly different if the TABLO-generated program is used.

[199] Suppose that imports of commodity 2 were originally 100 units of volume. Then, after a 25.0% increase they would be 125 units. Thus the reversal is a decrease of [25/125]*100=20.0%. Similarly for commodity 1.

For the data, you can use the GEMPACK program CMPHAR (see section 4.2 of GPD-4) to compare the post-simulation data MOTQ1DR.UPD after the reversal simulation with the original data MO.DAT. You will find that they agree to at least 3-4 figures. You should also compare the supplementary TRQ data, that is compare MOTQ1DRQ.UPD with the original MOTRQ.DAT. [These are both text files.] Again you will find good agreement.

16.8.3 Example: MOTR2 (Alternative Form of Miniature ORANI with Tariff-Rate Quotas)

MOTR2.TAB is the same as MOTRQ.TAB except that an alternative form of the COMPLEMENTARITY statement is used.

```
COMPLEMENTARITY
   (Variable = TIMP,
    Lower_Bound = TIMP_INQ,
    Upper_Bound = TIMP_OVQ)   TRQ2
  (All,i,COM) 1 - XIMP_RATIO(i) ;
```

In this alternative form, the complementarity variable is TIMP, the actual power of the import tariff. The lower and upper bounds are accordingly different from those in MOTRQ.TAB. In MOTR2.TAB the lower bound is TIMP_INQ (the in-quota power of the tariff) and the upper bound is TIMP_OVQ (the power of the tariff which applies if imports are over-quota).

The complementarity expression is the same as in MOTRQ.TAB.

16.8.3.1 Example Simulations with MOTR2 – MOT21C.CMF and MOT21D.CMF

These are very similar to the corresponding cases for MOTRQ.TAB You should run them and compare the results with those using MOTRQ.TAB. [First use AnalyseGE to look at the pre-sim and post-sim values of the complementarity variable and expression. Then look at some simulation results via ViewSOL.]

16.8.4 Example: G5BTRQ (GTAP with Bilateral Tariff-Rate Quotas)

The various files referred to here can be found in the file **G5TRQ.ZIP** which should be in the EXAMPLES subdirectory supplied with the GEMPACK software.

The files needed are:

- G5BTRQ.TAB, the TABLO Input file for GTAP with bilateral tariff-rate quotas.
- G5BTRQ.STI the condensation file for G5BTRQ.TAB .
- Standard data files for GTAP, SET6x4v5.HAR, DAT6x4v5.HAR and PAR6x4v5.HAR .
- Extra files with Bilateral TRQ data, TRQ6x4v5.HAR and QSHR6x4.DAT .
- 1A5.CMF and 1B5.CMF, two of the Command files for this model.

A brief introduction to tariff-rate quotas can be found in section 16.8.2 above. Further background information can be found in GTAP Technical Paper No 18 [Elbehri and Pearson (2000), which we refer to here as **TP18**].

16.8.4.1 G5BTRQ.TAB

This TAB file is very similar to the file GTAP5TRQ.TAB described in TP18.

In G5BTRQ.TAB, a COMPLEMEMNTARITY statement is used to specify the behaviour required for bilateral tariff-rate quotas. The COMPLEMENTARITY statement in G5BTRQ.TAB is

```
COMPLEMENTARITY
   (Variable = TMSTRQ,
    Lower_Bound = 1,
    Upper_Bound = TMSTRQOVQ)    TRQ
  (All,i,TRAD_COMM)(All,r,REG)(All,s,REG)
    1 - QXSTRQ_RATIO(i,r,s) ; ! The complementarity expression !
```

This Complementarity means:

Either TMSTRQ = 1	and	1 – QXSTRQ_RATIO > 0 [in-quota],
or 1 <= TMSTRQ <= TMSTRQOVQ	and	1 – QXSTRQ_RATIO = 0 [on-quota],
or TMSTRQ = TMSTRQOVQ	and	1 – QXSTRQ_RATIO < 0 [over-quota]

The condensation Stored-input file G5BTRQ.STI (for running TABLO) is supplied in G5TRQ.ZIP. You can switch between GEMSIM and TABLO-generated program output by switching between "pgs" and "wfp" near the end of this Stored-input file.

The "B" in the name of this stands for "Bilateral" to distinguish this from the Global TRQ case (where "G" is used – see section 16.8.6 below).

16.8.4.2 Connections Between Complementarity and Rest of TAB File

We have built G5BTRQ.TAB by adding a module to a standard version of the TABLO Input file GTAP.TAB for the GTAP model. In fact G5BTRQ.TAB is based on Version 5.0 (May 2000) of GTAP.TAB since that was the version of GTAP.TAB used as the basis of the TABLO Input files in TP18. In G5BTRQ.TAB, the TRQ module is added after the basic GTAP model and before the Welfare Decomposition and Terms of Trade modules. The TRQ module added to Version 5.0 of GTAP.TAB could easily be added to the current version of GTAP.TAB (Version 6.1, August 2001). Few, if any, changes would be needed.

As explained earlier (see section 16.3.5), a COMPLEMENTARITY statement must be expressed using levels variables. This may seem to be a problem when the COMPLEMENTARITY statement is being added at the end of a TAB file such as GTAP.TAB in which there are mainly (or exclusively) linear variables (changes and percentage changes) and very few (if any) levels variables.

This problem is easily solved, as we explain here. The idea is to add levels variables whose linearized versions are equal to linearized variables already in the model, and to add equations which link the two together (see also section 4.14.2 of GPD-2).

This is best explained and understood in the context of a concrete example. We describe below a couple of these linking equations from G5BTRQ.TAB.

One of the important relations is:

TMS(i,r,s) = TMSTRQ(i,r,s) * TMSINQ(i,r,s)

Since TMSTRQ and TMSINQ are introduced in the TRQ module as levels variables, it is natural to want to write this down as a levels equation. But in the GTAP.TAB being added to, the variable tms appears only as a linear, percentage-change variable, not as a levels variable. Introducing a levels version (which we call TMS_L to distinguish it from the "tms" already in GTAP.TAB) is done as follows:

```
VARIABLE (Levels, Linear_var = tms)
  (all,i,TRAD_COMM)(all,r,REG)(all,s,REG)        TMS_L(i,r,s)
 # Total import tariff for TRQ commodity i from r into s. # ;
Formula (Initial) (all,i,TRAD_COMM)(all,r,REG)(all,s,REG)
    TMS_L(i,r,s) =  VIMS(i,r,s)/VIWS(i,r,s) ;
Formula & Equation E_TMSTRQ (all,i,TRAD_COMM)(all,r,REG)(all,s,REG)
    TMSTRQ(i,r,s) = TMS_L(i,r,s) / TMSINQ(i,r,s) ;
```

The "Linear_var=tms" qualifier (see sections 2.2.2 and 3.4 of GPD-2) in the declaration of the new levels variable TMS_L indicates that the linear variable associated with TMS_L is just the variable tms which is already in the model. [That is, the variable tms indicates the percentage change in TMS_L.]

The initial values for TMS_L are set up via the Formula(Initial) shown. The link between the levels variables TMS_L, TMSINQ and TMSOVQ is made via the final Formula & Equation shown. This is written as shown since the Formula part is used to give the pre-simulation values for TMSTRQ.

Note that, prior to the introduction of the LINEAR_VAR= qualifier, the connection between the new levels variable TMS_L and the linear variable tms already in the TAB file would have been made as follows.

```
VARIABLE (levels)
  (all,i,TRAD_COMM)(all,r,REG)(all,s,REG)        TMS_L(i,r,s)
 # Total import tariff for TRQ commodity i from r into s. # ;
Equation (Linear) E_TMS_L
(all,i,TRAD_COMM)(all,r,REG)(all,s,REG) p_TMS_L(i,r,s) = tms(i,r,s) ;
```

The declaration of TMS_L here automatically generates the associated percentage-change linear variable p_TMS_L (see section 2.2.2 of GPD-2). The equation E_TMS_L says that p_TMS_L and tms are the same. [If you look in GTAP5TRQ.TAB in Elbehri and Pearson (2000), you will find these equations since that TAB file was written while Release 7.0 of GEMPACK was current. At that time, LINEAR_VAR= qualifiers were not available.] We strongly recommend that you use LINEAR_VAR= qualifiers to simplify linking code.

If you look in G5BTRQ.TAB near the start of the Tariff-Rate Quota module added at the end, you will see other linking equations.

The general procedure for providing equations to link a levels module (at the end of a TAB file) to a linear model is also described in detail in the document C12-01

"Adding Accounting-Related Behaviour to a Model Implemented Using GEMPACK"

Harrison and Pearson (2002).

16.8.4.3 Example Simulations with G5BTRQ

Any of the example simulations and applications described in **TP18** could be carried out using G5BTRQ. We refer you to that Technical Paper for more details about these simulations, the rationale behind them and the construction of the starting data base.

We have supplied Command files for several example simulations with G5BTRQ.TAB. We discuss these briefly below.

We encourage you to carry out all of these simulations if you are interested in modelling tariff-rate quotas.

Checking the Simulation Results

When you carry out one of these simulations, you should check the results in much the same way as you checked the results of the simulations in sections 16.4 and 16.8.2 above. In particular,

- you should use AnalyseGE to look at the pre-sim and post-sim states and values of the complementarity variables and expressions.
- you should look at the log file to follow through the state changes during the approximate run and to look at the other state reports. We give a detailed example of this in section 16.8.4.4 below.
- you should look at the updated data (especially the updated supplementary TRQ data) to check that the values there are consistent with the post-simulation states.

When checking a reversal simulation (for example, 1A5R.CMF and 1B5R.CMF), you should follow the methods described in the discussion of the reversal MOTQ1DR.CMF simulation in section 16.8.2.3 above to check that

- states and values of complementarity variables and expressions return to their original value.
- simulation results are reversed
- the updated data after the reversal simulation is essentially identical to the data before the original simulation.

Increasing the In-quota Tariff – Command files 1A5.CMF and 1B5.CMF

These are simulations in which the in-quota tariff on sugar imported from Africa (AFR) to the United States (USA) is increased. These are what are called Cases 1A and 1B in TP18. In 1B5, the triple (sugar,AFR,USA) moves from state 3 (over-quota) to state 1 (in-quota).

Reversals – Command files 1A5R.CMF and 1B5R.CMF

The reversal 1B5R.CMF of case 1B is especially interesting. The shock in the Case 1B simulation is relatively large (a doubling of the in-quota tariff for sugar from AFR to USA) so this simulation is relatively nonlinear. Accordingly the reversal simulation has to "work quite hard" to get back to the original (that is, before Case 1B is run) over-quota state 3. This is especially so since the original state of (sugar,AFR,USA)[200] is over quota (state 3) but not all that far over quota since the original import volume is 1.057 times the quota volume.

If you try the reversal 1B5R.CMF with 20 steps in the approximate run, the reversal simulation fails (even though the accurate run converges well). [To carry out 1B5R with just 20 Euler steps, change the relevant statement in 1B5R.CMF to "complementarity steps_approx_run = 20 ;".] The LOG file shows various problems. The most obvious ones are that, at the end of the accurate run,

- the post-simulation value of TMSTRQ(sugar,AFR,USA) is 1.844 which exceeds the upper bound of 1.8199 [the value of TMSTRQOVQ(sugar,AFR,USA)] specified in the COMPLEMENTARITY statement.
- the post-sim state for triple (sugar,AFR,USA) is 2 (on-quota) after the approximate run while it is 3 (over-quota) after the accurate run.

Thus the simulation fails with 20 Euler steps. Similar problems occur if you take 40 Euler steps (when again the post-sim state for (sugar,AFR,USA) is 2 after the approximate run instead of the desired state 3). However these problems all disappear if you take 50 Euler steps.

Reversal of 1B5 with Automatic Accuracy – Command file 1B5RAA.CMF

Having to reach the correct post-sim state in a relatively nonlinear simulation like this in a single Euler calculation is quite demanding. It is for this reason that it may be better to allow the software to break the simulations (approximate and accurate) into several subintervals (of possibly different lengths). This is what happens with user-specified accuracy, which you have when you put the statement "automatic accuracy = yes ;" in your Command file (see section 16.7.4 for more details). You can see the effects of this if you carry out the reversal simulation using Command file **1B5RAA.CMF**. [In this, only 20 Euler steps are specified for the approximate run. The software takes 2 subintervals, each taking 20 Euler steps for the approximate run, to complete the run. Although the total CPU time may be greater than for a single subinterval with 50 Euler steps

[200] The triple (i,r,s) relates to imports of commodity i from region r to region s. Thus (sugar,AFR,USA) refers to imports of sugar from AFR into USA.

in the approximate run, you may waste a lot of time experimenting before you realise that 50 steps are required in that case.]

Reversal of 1B5 with Automatic Accuracy again – Command file 1B5RAAX.CMF

This is the same as 1B5RAA.CMF except that one subinterval is specified to start with. The attempt to do the whole simulation in one subinterval fails because the post-sim state for (sugar,AFR,USA) is 3 after the accurate simulation but 2 after the approximate simulation. The subinterval length is reduced and the simulation then proceeds ok.

16.8.4.4 Reporting State Changes – Command file 1B5.CMF

Whenever you carry out a simulation in which one or more complementarities may change state, you should check the state changes, as reported in the log file of the run. We illustrate this below by following through in detail the reports about states and state changes in the log file from the 1B5.CMF simulation. [You can do the same for the other simulations with this and other models involving complementarities.]

In this simulation, the in-quota power of the tariff on (sugar,AFR,USA) is doubled and the ratio between over-quota and in-quota powers remains fixed. This moves the triple (sugar,AFR,USA) from over-quota to in-quota.

There are 20 Euler steps in the initial approximate simulation followed by an accurate simulation using the Gragg method with 8, 12, 16 steps:

```
Method = Gragg ;
Steps = 8  12  16 ;
complementarity steps_approx_run = 20 ;
```

[The "steps_approx_run" statement is a way of telling the software how many Euler steps to take in the approximate run. It is optional. See section 16.6 for more details.]

It is instructive to follow the state changes reported during this run.

In the log file **1b5.log** from this run, you can see the approximate run (a 20-step Euler). At the beginning the program reports that 136 of the 144 triples (i,r,s)[201] are in state 1 (in-quota) and that 8 are in state 3 (over quota). During step 1 of the 20 Euler steps, the program reports that (sugar,AFR,USA) moved from state 3 to state 2 (on quota). During step 16 it reports that (sugar,AFR,USA) moved from state 2 to state 1. After the approximate run it tells you that 1 triple moved from state 3 to state 1 and that the other 143 remained in their pre-simulation states.

[To see the state reports, search for "state" in the log file.]

- The pre-simulation state report is as follows:

```
[Complementarity TRQ. Numbers initially in states
 1, 2 and 3 are 136, 0 and 8 respectively.]
```

- The program will report changes in state as they occur throughout the approximate simulation. For example, there is a state change from state 3 to state 2 during Euler step number 1, as reported below:

```
[Complementarity TRQ("Sugar","AFR","USA"):
  State change from 3 to 2 during this step.
   Occurs after 0.25966060 part of the step.]
(Redoing this step to change state more gradually.)
```

- Whenever a state change occurs during a step, the step is repeated with a shorter step length to ensure that the state change occurs near the end of this (shorter) step. [The reasons for this are set out in section 16.7.3.] Because of the above state change during step 1, this step is carried out again. The state change report in this repeated step is as follows:

```
[Complementarity TRQ("Sugar","AFR","USA"):
  State change from 3 to 2 during this step.
   Occurs after 0.99961501 part of the step.]
```

- Then later there is another state change from state 2 to state 1 during Euler step 17:

```
--->  Beginning pass number 17 of 20-pass calculation.
[Complementarity TRQ("Sugar","AFR","USA"):
  State change from 2 to 1 during this step.
   Occurs after 0.79648900 part of the step.]
```

[201] The triple (i,r,s) relates to imports of commodity i from region r to region s.

```
(Redoing this step to change state more gradually.)
```

- This step is also repeated (with a shorter length). The reported state change during this repeated step number 17 is:

```
(Beginning the update after pass 17 of this 20-pass calculation.)
[Complementarity TRQ("Sugar","AFR","USA"):
  State change from 2 to 1 during this step.
   Occurs after 0.99987364 part of the step.]
```

- At the end of the approximate simulation just before starting the accurate Gragg simulation there is a report of state changes which occurred during the approximate run, and which are therefore expected during the accurate simulation :

```
===> Beginning accurate complementarity calculation.

[Processing complementarity TRQ.]
 [Are 0 changing from state 1 to state 2, and
      0 changing from state 1 to state 3.]
 [Are 0 changing from state 2 to state 1, and
      0 changing from state 2 to state 3.]
 [Are 1 changing from state 3 to state 1, and
      0 changing from state 3 to state 2.]
 [Numbers remaining unchanged in states 1,2 and 3 are
   136, 0 and 7 respectively.]
 [Total numbers in post-sim states 1,2 and 3 are
   137, 0 and 7 respectively.]

--->  Beginning pass number 1 of 9-pass calculation.
```

16.8.5 Alternative Form of the Complementarity – G5BTR2.TAB

The files needed are:

- G5BTR2.TAB, the TABLO Input file for GTAP with Bilateral Tariff-Rate Quotas.
- G5BTR2.STI the condensation file for G5BTRQ.TAB .
- Standard data files for GTAP, SET6x4v5.HAR, DAT6x4v5.HAR and PAR6x4v5.HAR .
- Extra files with Bilateral TRQ data, TRQ6x4v5.HAR and QSHR6x4.DAT .

The Command files for the G5BTRQ model can be used with this model by changing the name of the auxiliary files to ***g5btrq2*** instead of ***g5btrq***.

This TAB file is the same as G5BTRQ.TAB except that an alternative form of the complementarity statement is used. This version has TMS_L [the levels value of the actual total import tariff (including any due to the tariff-rate quota)] as the complementarity variable instead of TMSTRQ (the extra power of the tariff due to the tariff-rate quota). The lower and upper bounds in this second case are TMSINQ (the in-quota power) and TMSOVQ (the total over-quota power) respectively. The complementarity statement in G5BTR2.TAB is:

```
COMPLEMENTARITY (Variable = TMS_L,
      (Lower_Bound = TMSINQ,      Upper_Bound = TMSOVQ)    TRQ2
  (All,i,TRAD_COMM)(All,r,REG)(All,s,REG)
    1 - QXSTRQ_RATIO(i,r,s) ; ! The complementarity expression !
```

This second form of the complementarity means:

Either	TMS_L = TMSINQ	and	1 – QXSTRQ_RATIO > 0 [in-quota],
or	1 <= TMS_L <= TMSOVQ	and	1 – QXSTRQ_RATIO = 0 [on-quota],
or	TMS_L = TMSOVQ	and	1 – QXSTRQ_RATIO < 0 [over-quota]

The condensation Stored-input file G5BTR2.STI (for running TABLO) is supplied in G5TRQ.ZIP. You can switch between GEMSIM and TABLO-generated program output by switching between "pgs" and "wfp" near the end of this Stored-input file.

16.8.5.1 Example Simulations with G5BTR2

Any of the simulations with G5BTRQ.TAB can be carried out with G5BTR2.TAB. We supply only the Command file **1B5R2.CMF** carries out the reversal of Case 1B (see section 16.8.4.3 above) using G5BTR2.TAB instead of G5BTRQ.TAB.

We suggest that you carry out the 1B5R2.CMF simulation and check that the results are the same as for the 1B5R.CMF simulation which is based on G5BTRQ.TAB.

In this reversal 1B5R2.CMF simulation, the post-simulation state of the triple (sugar,AFR,USA) is 3 (over-quota). Thus, in the accurate simulation, the complementarity variable must be shocked so that it becomes equal to the upper bound. With the alternative form of the complementarity in G5BTR2.TAB (see above), the upper bound TMSOVQ is endogenous. Hence this is an example where the variable TRQ2@U (set equal to the difference between the complementarity variable TMS_L and the upper bound – see Case 3(b) in section 16.7.1) is made exogenous and shocked (to zero).

16.8.6 Example: G5GTRQ (GTAP with Global and Bilateral TRQs)

With a bilateral TRQ, the trigger is the bilateral import volume. In contrast, a **global TRQ** is one which is triggered by aggregate imports (from all exporting regions).

The files needed are:

- G5GTRQ.TAB, the TABLO Input file for GTAP with global and bilateral tariff-rate quotas.
- G5GTRQ.STI the condensation file for G5GTRQ.TAB .
- Standard data files for GTAP, SET6x4v5.HAR, DAT6x4v5.HAR and PAR6x4v5.HAR .
- Extra file with supplementary global TRQ data, GTRQ6x4.DAT .

In G5GTRQ.TAB is a

- bilateral TRQ for commodity sugar, and
- a global TRQ for commodity **othag** (other agriculture).

[It would be easy to modify this TAB file to apply to different commodities.]

You may like to look in G5GTRQ.TAB to see how the global TRQ is specified.

In the supplementary global TRQ data supplied (see file **GTRQ6X4.DAT**), imports of othag into the 4 regions ASI (Asia), LAM (Latin America), AFR (Africa) and ROW (Rest-of-World) are over quota while imports into the remaining 2 regions USA and E_U (Europe) are in quota. The full extra power of the tariff TMTRQOVQ due to the global TRQ is 8.0 for USA and E_U and is 1.4 for the other 4 regions. Imports into USA and E_U are assumed to be 0.8 of the quota volume while imports into the other 4 regions are assumed to be 1.2 times the quota volume.

The condensation Stored-input file G5GTRQ.STI (for running TABLO) is supplied in G5TRQ.ZIP. You can switch between GEMSIM and TABLO-generated program output by switching between "pgs" and "wfp" near the end of this Stored-input file.

The "G" in the name of this stands for "Global" to distinguish this from the Bilateral TRQ case (where "B" is used – see section 16.8.4 above).

16.8.6.1 Example Simulations with G5GTRQ

Below we describe briefly the example simulations provided with the GEMPACK examples.

We suggest that you carry out all these simulations and that you check the results following the general guidelines given above under the heading "Checking the Simulation Results" in section 16.8.4.3.

Bilateral and Global TRQ combined – Command file G2B5.CMF

In this Command file, the Case 1B bilateral TRQ shock (see 1B5.CMF above) is given. There is also a 25% increase in the global TRQ quota volume for imports of othag into Africa.

In the pre-simulation data, (othag,AFR) is in state 3 with respect to the global TRQ. The effect of these shocks is to bring (othag,AFR) on-quota with respect to the global TRQ and to bring (sugar,AFR,USA) in-quota with

respect to the bilateral TRQ on sugar. If you carry out this simulation, you can see from the LOG file approximately when these state changes occur during the approximate run. The state changes for the whole simulation are reported at the end of the whole run.

You can find interesting information about the 2 complementarities via AnalyseGE.

Reversal of G2B5 – Command file G2B5R.CMF

As with 1B5R (see above), 40 Euler steps are not sufficient for the approximate run, but 50 steps are sufficient. [Again automatic accuracy would be a good option.]

When you check that the reversal has worked, and are looking to compare the updated data after the reversal simulation with the original data, you should remember to compare the supplementary data files as well as the main GTAP data file. For example, you should compare the updated supplementary global TRQ data in file G2B5RG.UPD with the original data in file GTRQ6X4.DAT (which you can do in a text editor since they are text files).

16.8.7 Example: G94-XQ (GTAP94 with Bilateral Export Quotas)

The various files referred to here can be found in the file **G94-QUO.ZIP** which should be in the EXAMPLES subdirectory supplied with the GEMPACK software.

The files needed are:

- G94-XQ.TAB, the TABLO Input file for GTAP94 with bilateral export quotas.
- G94-XQ.STI the condensation file for G94-XQ.TAB .
- Standard data files for GTAP94, SET2-01.HAR, DAT2-01.HAR and PAR2-01.DAT .
- Extra file with supplementary data, XQ2-01.DAT .
- Command files XQ-A2.CMF, XQ-A4.CMF, XQ-A6.CMF and XQ-P2.CMF.

This example is supplied as an updated version of the bilateral export quota model and applications supplied with Bach and Pearson (1996) which is GTAP Technical Paper No. 4 [**TP4**]. The Technical Paper TP4 described an early implementation of bilateral export and import quotas with GTAP and GEMPACK.

16.8.7.1 Additional Export Quota Data XQ2-01.DAT

Extra data is required to indicate the pre-simulation state of the quotas. As in TP4, this extra data is for 3 tradeable commodities (food, manufactures, services) and three regions USA, EU and ROW. The extra quota data is set up so that no quotas are binding. Exports of manufactures "mnfcs" from ROW to USA and EU are 0.98 of the quota volume (that is, these two quotas are nearly binding). In all other cases, exports are 0.2 of the quota volume (that is, the quotas are far from binding).

In TP4 the extra data consisted of VXQD (values of exports by destination at prices which include normal export taxes or subsidies, but do not include the extra tax due to the quota) and QXS_RATIO values (the ratio of current export volume to the quota volume). In G94-XQ.TAB we have chosen to hold TXS_QUOTA (the extra power of the export tax due to the quota) instead of the VXQD values. Because no quotas are binding in this extra data, all TXS_QUOTA values are 1.

16.8.7.2 G94-XQ.TAB

This is version 2.2a (August 1995) of GTAP94.TAB with export quotas added. The TAB file statements to implement export quotas are at the end of that file. Similar (probably identical) statements could be added to more recent versions of GTAP.TAB to do the same job.

The export quota module in G94-XQ.TAB file is shown below. Noteworthy features are:

- Only two levels variables TXS_QUOTA and QXS_RATIO are added for the complementarity. These must be levels variables because the COMPLEMENTARITY statement requires levels variables (see section 16.3.5).
- The only other variables required to link to the rest of the model are qxs_quota and pxs. Each of these has been added as a linear (percentage-change) variable [although these could have been added as levels variables].
- The accounting identities linking prices and ratios have been put in as Linear equations called E_QXS_RATIO and E_PFOB.
- The EXPRICES equation from the standard model must be altered as shown. [The version in the top of the file has been commented out.]
- The choice of which extra quota-related data to read (see section 16.8.7.1) affects the way the equations are written down. [If VXQD were read, as in TP4, probably some extra levels prices would have been needed.]
- Having only two new levels variables means that the code to link the quota code to the rest of the model (see section 4.14.2 of GPD-2 and section 16.3.5 above) is extremely simple in this case.

Also shown below are the statements used to calculate and report the changes in the associated quota rents (see the equation E_QXS_RENT_RAT).

```
! Addition for export quotas in G94-XQ.TAB !
VARIABLE (Levels) (all,i,TRAD_COMM)(all,r,REG)(all,s,REG) TXS_QUOTA(i,r,s)
  # extra export tax, due to quota, in r on good i bound for region s # ;
VARIABLE (Levels) (all,i,TRAD_COMM)(all,r,REG)(all,s,REG) QXS_RATIO(i,r,s)
   # Ratio between volume of exports of i from r to s and import quota # ;
VARIABLE (Linear) (all,i,TRAD_COMM)(all,r,REG)(all,s,REG) qxs_quota(i,r,s)
   # Volume export quota on exports of i from r to s # ;
VARIABLE (Linear) (all,i,TRAD_COMM)(all,r,REG)(all,s,REG) pxs(i,r,s)
  # price of i supplied from r to s inclusive of normal export tariffs #;
  ! But exclusive of export quota tariff equivalents. May be less than
  PMS(i,r,s) in the levels if the export quota on i from r to s is binding !

FILE (TEXT) EXP_QUOTA  # File containing export quota data # ;

READ  QXS_RATIO  FROM FILE EXP_QUOTA ;
READ  TXS_QUOTA  FROM FILE EXP_QUOTA ;

Equation (Linear)  E_QXS_RATIO # Links export volume, quota and ratio #
  (all,i,TRAD_COMM)(all,r,REG)(all,s,REG)
  ! Levels equation is: QXS_RATIO=QXS_L/QXS_QUOTA  !
     qxs(i,r,s) = p_QXS_RATIO(i,r,s) + qxs_quota(i,r,s) ;

EQUATION (Linear) E_PFOB
  # Links dom price of exports with usual and quota-related export tariffs #
(all,i,TRAD_COMM)(all,r,REG)(all,s,REG)
    pfob(i,r,s) = pxs(i,r,s) + p_TXS_QUOTA(i,r,s) ;

! Next equations are modified from the usual GTAP94.TAB version
  If this quota addition is used, the original equation (EXPRICES)
  should be deleted or excluded with strong comment marks.
 This equation links agent's and world prices. In addition to tx we have txs
 which embodies both production taxes (all s) and export taxes (r not equal
 to s) (HT#27)!

EQUATION EXPRICES (all,i,TRAD_COMM)(all,r,REG)(all,s,REG)
pxs(i,r,s) = pm(i,r) - tx(i,r) - txs(i,r,s) ;

COMPLEMENTARITY
  (Variable = TXS_QUOTA,  Lower_Bound = 1)  EXPQUOTA
  (all,i,TRAD_COMM)(all,r,REG)(all,s,REG)      1 - QXS_RATIO(i,r,s) ;

VARIABLE (Levels, Change)
    (all,i,TRAD_COMM)(all,r,REG)(all,s,REG)      QXS_RENT_RAT(i,r,s)
  # Ratio of quota rent to pre-quota value of exports for each commodity #;
 ! Here pre-quota value is taken as the value including normal export
     tariffs - that is, QXS_L*PXS_L.
   Thus, in the levels,
         QXS_RENT_RATIO_L= [QXS_L*(PFOB_L-PXS_L)]/[QXS_L*PXS_L]
                                = [PFOB_L/PXS_L] - 1
                                = TXS_QUOTA - 1   !

Equation (Levels) E_QXS_RENT_RAT (all,i,TRAD_COMM)(all,r,REG)(all,s,REG)
    QXS_RENT_RAT(i,r,s) = TXS_QUOTA(i,r,s) - 1 ;
```

16.8.7.3 Example Simulations with G94-XQ

Those supplied are the accurate versions of the simulations XQ-A2.CMF, XQ-A4.CMF, XQ-A6.CMF and XQ-P2.CMF from **TP4**. All the same results as in **TP4**.

We suggest that you carry out all these simulations and that you check the results following the general guidelines given above under the heading "Checking the Simulation Results" in section 16.8.4.3.

We also suggest that you read the description of these simulations in TP4 and check that the results here are as reported there.

Decreasing the Quota – Command file XQ-A2.CMF

The shock in this simulation will reduces the quota for exports of mnfcs from ROW to USA. This makes it binding.

```
shock qxs_quota("mnfcs","ROW","USA") = -10;
```

Reversal of the XQ-A2.CMF Simulation – Command file XQ-A4.CMF

This simulation reverses the XQ-A2.CMF simulation.

```
shock qxs_quota("mnfcs","ROW","USA") = 11.111111 ;
```

Increase One Quota and Decrease Another Quota – Command file XQ-A6.CMF

This simulation starts from the post-simulation data after simulation XQ-A2. In simulation XQ-A2, the quota on (mnfcs,ROW,USA) was decreased by 10% which makes it binding.

The shocks in XQ-A6.CMF are to

- return the quota on (mnfcs,ROW,USA) to its original value (which should make it non-binding),
- decrease the quota on (mnfcs,ROW,EU) by 10 percent (which should make it binding).

It is instructive to look at the information which AnalyseGE provides if you right click on the COMPLEMENTARITY statement (in the TABmate form) and then select item **Show State and Variable Values.** If you select the drop-down boxes in the top right-hand corner of the ViewHAR form to show **mnfcs**, **ROW**, **All REG**, **All $CPINFO** respectively, you will see the following matrix (in which the row and column totals have been suppressed since they are of no value here).

	STATE-PRE	STATE-POST	VAR-PRE	VAR-POST	EXP-PRE	EXP-POST	LB-PRE	LB-POST
USA	2	1	1.037662	1	0	0.006597	1	1
EU	1	2	1	1.037282	0.007942	0	1	1
ROW	1	1	1	1	0.798572	0.798366	1	1

Look first at the USA row. This shows that the triple (mnfcs,ROW,USA) has changed from state 2 (STATE-PRE) where the quota is binding to state 1 (STATE-POST) where the quota is not binding. The values of the complementarity variable (TXS_QUOTA) confirm this. The pre-simulation value is 1.037662 (VAR-PRE) while the post-simulation value (VAR-POST) appears to be exactly 1. The values of the complementarity expression (1 – QXS_RATIO) also confirm this. The pre-simulation value (EXP-PRE) is exactly zero while the post-simulation value (EXP-POST) is 0.006597 which means that exports are approximately 0.993403 of the quota volume (that is, the quota is nearly binding). The last two columns (LB-PRE and LB-POST) shown the pre- and post-simulation values of the lower bound (both exactly 1).

You should also look at the EU and ROW rows. Check that all information in the EU row is consistent with the triple (mnfcs,ROW,EU) moving from non-binding to binding. Have exports of (mnfcs,ROW,ROW) moved closer to or further away from the (non-binding) quota?

16.8.8 Example: G94-IQ (GTAP94 with Bilateral Import Quotas)

The various files referred to here can be found in the file **G94-QUO.ZIP** which should be in the EXAMPLES subdirectory supplied with the GEMPACK software.

This example is supplied as an updated version of the bilateral import quota model and applications supplied with Bach and Pearson (1996) which is GTAP Technical Paper No. 4 [**TP4**]. The Technical Paper TP4 described an early implementation of bilateral export and import quotas with GTAP and GEMPACK.

In TP4 bilateral import quotas are specified in the TAB file GTAP33IQ.TAB for all commodities, though the applications only relate to quotas on food imports. In G94-IQ.TAB the quotas are only specified for food (though it would be easy to add other commodities).

The files needed are:

- G94-IQ.TAB, the TABLO Input file for GTAP94 with bilateral import quotas.
- G94-IQ.STI the condensation file for G94-IQ.TAB .
- Standard data files for GTAP, SET2-01.HAR, DAT2-01.HAR and PAR2-01.DAT .
- Extra file with bilateral import data, IQ-FOOD.DAT .
- Command files IQ-B1.CMF and IQ-B3.CMF, also IQ-QUO1.CMF .

16.8.8.1 Additional Export Quota Data IQ-FOOD.DAT

Extra data is required to indicate the pre-simulation state of the quotas. As in TP4, this extra data is for 3 tradeable commodities (food, manufactures, services) and three regions USA, EU and ROW. The extra quota data is set up so that all quotas are just binding – that is, they bind but the extra power of the tariff due to the quotas is still 1.

In TP4 the extra data consisted of VIQS (values of imports by source at prices which include normal import taxes or subsidies, but do not include the extra tax due to the quota) and QIS_RATIO values (the ratio of current import volume to the quota volume). In G94-XQ.TAB we have chosen to hold TMS_QUOTA (the extra power of the import tax due to the quota) instead of the VIQS values. Because all quotas are just binding in this extra data, all TMS_QUOTA values are 1.

16.8.8.2 G94-IQ.TAB and Example Simulations

You may wish to look at the import quota implementation in G94-IQ.TAB. The relevant statements are near the bottom of the file. We have adopted a similar strategy to that in G94-XQ.TAB of adding the minimum number of levels variables in order to simplify the links with the rest of the model.

We suggest that you carry out all these simulations and that you check the results following the general guidelines given above under the heading "Checking the Simulation Results" in section 16.8.4.3.

We also suggest that you read the description of these simulations in TP4 and check that the results here are as reported there.

The applications **IQ-B1.CMF** and **IQ-B3.CMF** can be run with this model. All quotas start as binding which is classed as state 2 by the complementarity software. Because these quotas are just binding, there can be a large number of state changes at the start of any simulation (since a tiny move towards the quota being not binding will result in a state change). You can see these state changes if you look in the log files from these simulations.

In the **IQ-B1.CMF** simulation, the tariff on food imports of food from USA and ROW into EU is decreased. Since these quotas were originally binding, they remain binding. There are no quantity changes and the only price change is a 10% fall in the imported price (including normal import tariff but excluding the tariff equivalent of the import quota) of these two imports [this price is variable **qis**]. There is an increase in the associated quota rents as you can see from the **c_QIS_RENT_RAT** results.

In the **IQ-B3.CMF** simulation, as well as the tariff reductions in IQ-B1.CMF (see above), the quota volumes on imports of food from USA and ROW into the EU are increased by 15 percent. These quotas remain binding (and there is as associated increase in these two import volumes of 15 percent – the variable **qxs**). However the quotas on imports of food from ROW into USA and from EU into itself become non-binding, as you can see when you check the results.

We include another example **IQ-QUO1.CMF**. In this example, quota volumes on food from both USA and ROW into EU are increased by 15 percent. Since all quotas are just binding in the pre-simulation data, the post-simulation equilibrium should be identical to the pre-simulation one except that these two quotas will be non-binding. Although not economically interesting, this simulation is interesting from a numerical point of view. After the approximate run, the post-simulation state for triple (food,EU,EU) is 1 (non-binding) whereas the state is 2 (binding) after the accurate run. Normally this would result in an error. But in this case, the software is able to tell that the post-simulation state after the accurate run (state 2) is also very close to state 1, so the software judges this to be ok. [In fact the post-sim state for this triple should be that the quota is just binding. It is somewhat arbitrary whether the software declares this to be state 1 or 2 since it is really on the corner in Figure 16.2b.]

16.9 Piecewise Linear Functions via Complementarities

In this section we show how various piecewise linear functions can be expressed accurately using Complementarity statements.

We begin with MAX and MIN. In view of section 16.3.6 above, it is not surprising that MAX and MIN can be expressed in terms of Complementarities.

We also include an example of a progressive income tax schedule in section 16.9.2. Other piecewise linear functions can be expressed similarly, as the examples in section 16.9.3 show.

16.9.1 Max and Min

16.9.1.1 Z = MAX(0,Y)

Firstly consider the function Z = MAX(0,Y) [See Figure 16.9.1.1 on the next page.]

This can be expressed via the Complementarity statement

```
COMPLEMENTARITY (Variable = Z, Lower_Bound = 0) <name>  Z – Y ;
```

The Complementarity Variable is Z, the Lower Bound is zero and the Complementarity Expression is Z – Y.

To see this, suppose that Z = MAX(0,Y). Firstly note that Z >= 0 always so that 0 is a lower bound for Z. There are two cases:

(i) Suppose Y >= 0. Then Z=Y and so Expression Z – Y=0. Hence in this case,
Variable Z >= Lower Bound 0 and Expression = 0. This is state 2 for the above Complementarity.

(ii) Suppose Y < 0. Then Z=0 so Variable is equal to Lower Bound. Also Expression Z – Y > 0. This is state 1 for the above Complementarity.

16.9.1.2 Z = MAX(X,Y)

Firstly note that Z = MAX(X,Y)

is the same as Z – X = MAX(0,Y – X).

We know from the section above how to express MAX(0,Y – X) using a Complementarity statement.
Hence Z = MAX(X,Y) can be expressed using an equation to introduce variable W=Z–X and then using the Complementarity

```
COMPLEMENTARITY (Variable = W, Upper_Bound = 0) <name>  W – (Y-X) ;
```

to express W=MAX(0,Y – X). Thus

Z = MAX(X,Y) can be expressed via the statements

```
VARIABLE (LEVELS) W  # Difference between Z and X # ;
EQUATION (LEVELS)  E_W   W = Z – X ;
COMPLEMENTARITY (Variable = W, Lower_Bound = 0) C_Z  W – (Y-X) ;
```

An alternative way of expressing Z=MAX(X,Y) can be found in section 16.9.3.5 below.

16.9.1.3 Z = MIN(0,Y)

This is very similar to MAX in section 16.9.1.1 above. You can see that

Z = MIN(0,Y) [see Figure 16.9.1.3 on the next page]

can be expressed via the Complementarity statement

```
COMPLEMENTARITY (Variable = Z, Upper_Bound = 0) <name> Z – Y ;
```

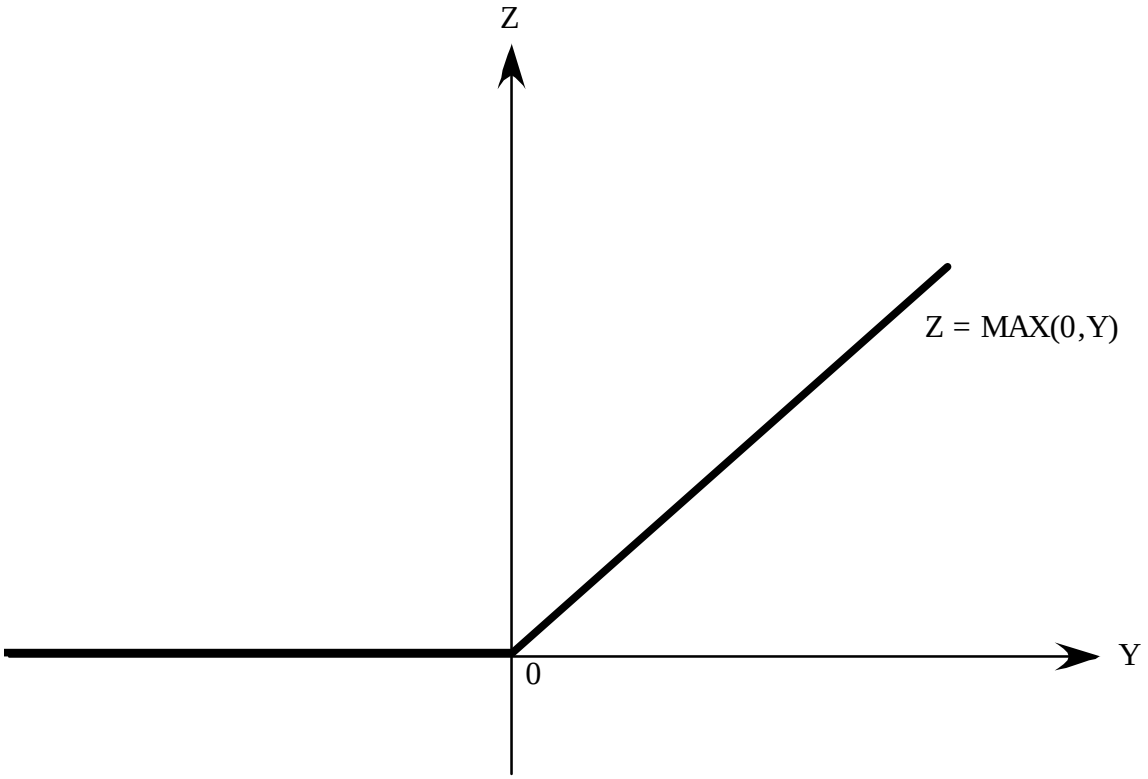


Figure 16.9.1.1 : Z = MAX(0,Y)

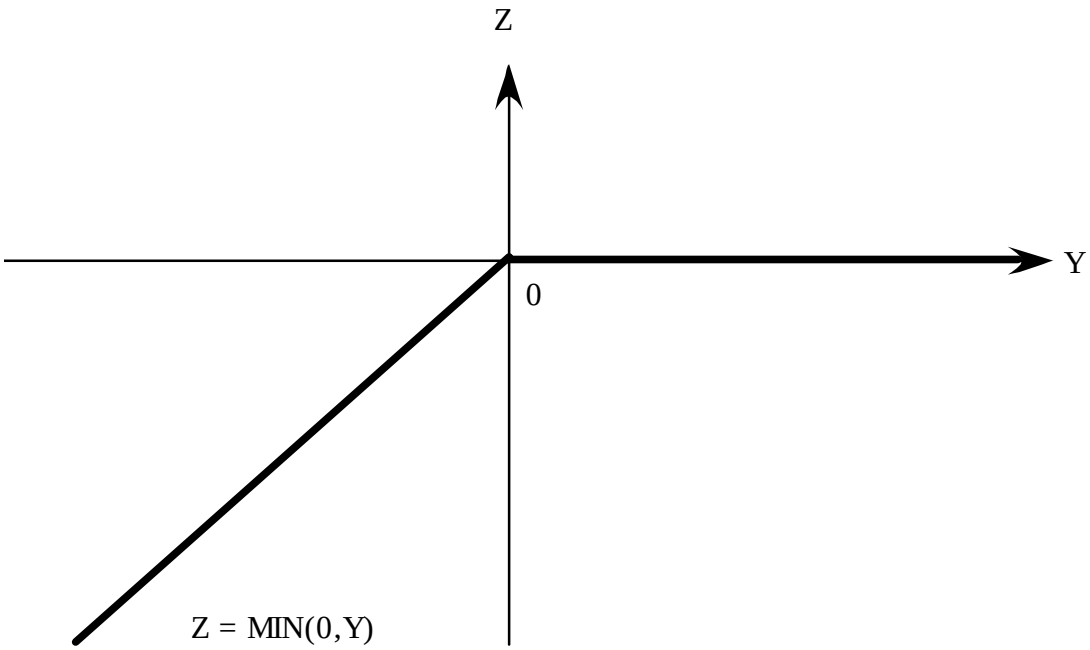


Figure 16.9.1.3 : Z = MIN(0,Y)

16.9.1.4 Z = MIN(X,Y)

Firstly note that Z = MIN(X,Y)

is the same as Z – X = MIN(0,Y – X).

Thus Z = MIN(X,Y) can be expressed via the statements

```
VARIABLE (LEVELS) W  # Difference between Z and X # ;
EQUATION (LEVELS)  E_W   W = Z – X ;
COMPLEMENTARITY (Variable = W, Upper_Bound = 0) C_Z  W – (Y-X) ;
```

An alternative way of expressing Z=MIN(X,Y) can be found in section 16.9.3.3 below.

16.9.2 Progressive Income Tax Schedule

Consider a progressive income tax schedule as follows:

- for the first $5000 of income, no tax is paid.
- for any income between $5,001 and $20,000, tax is paid at the rate of 10 cents in the dollar.
- for any income above $20,001, tax is paid at the rate of 20 cents in the dollar.

[See Figure 16.9.2 on the next page.]

The formula connecting INCOME and TAX is as follows:

TAX = 0	if INCOME <= 5000,
TAX = 0.1*(INCOME – 5000)	if 5000 <= INCOME <= 20000,
TAX = 0.1*15000 + 0.2*(INCOME – 20000)	if INCOME > 20000.

You can see that the first two parts of this can be expressed by saying that

TAX = MAX(0, 0.1*(INCOME – 5000)) .

Hence, if we introduce a new variable TAX2 via

TAX2 = MAX(0, 0.1*(INCOME – 5000)) ,

then you can see that the full rule above connecting INCOME and TAX can be expressed via

TAX = MAX(TAX2, 0.1*15000 + 0.2*(INCOME – 20000)) .

Thus (using sections 16.9.1.1 and 16.9.1.2 above) the piecewise linear connection between INCOME and TAX can be expressed in TABLO language as follows.[202]

```
VARIABLE (LEVELS) TAX2  # First 2 parts of tax schedule # ;

COMPLEMENTARITY (Variable = TAX2, Lower_Bound = 0)  C_TAX2
       TAX2 – 0.1*(INCOME – 5000) ;

VARIABLE (LEVELS) TAX3 # Difference between TAX and TAX2 # ;

EQUATION (LEVELS) E_TAX3       TAX3 = TAX – TAX2 ;

COMPLEMENTARITY (Variable = TAX3, Lower_Bound = 0)   C_TAX
      TAX3 – { [0.1*15000 + 0.2*(INCOME - 20000)] - TAX2 } ;
```

[202] For the second part of this, TAX = MAX(TAX2, <expression>), use the result in section 16.9.1.2 above. Replace Z there by TAX, X there by TAX2, Y there by <expression> and W there by TAX3.

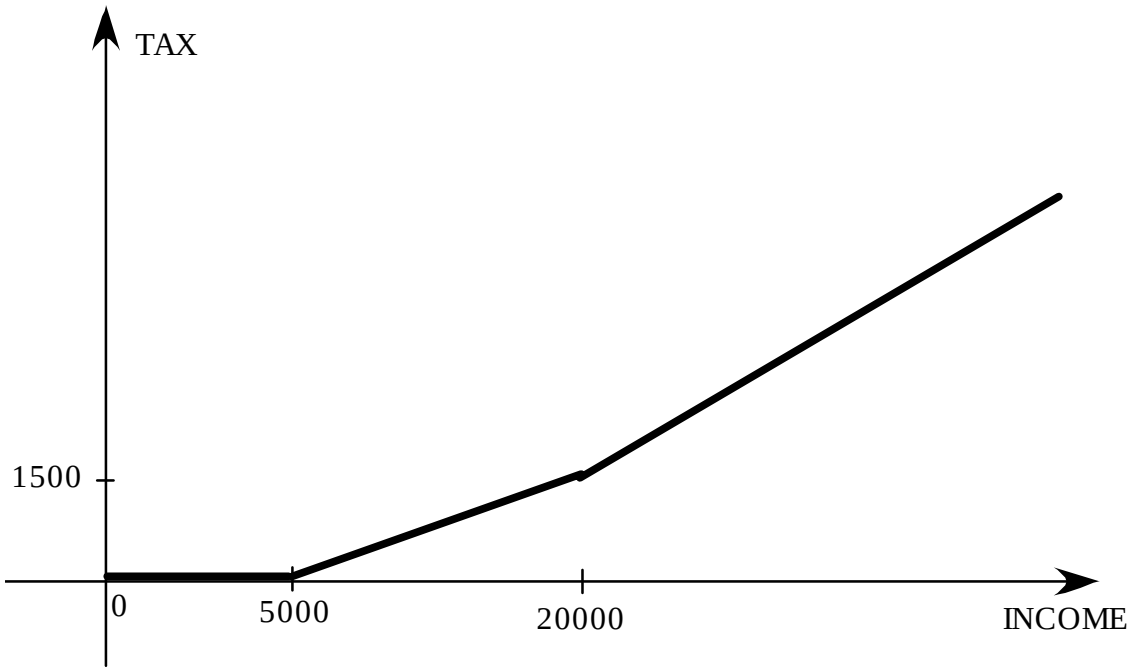


Figure 16.9.2 : Progressive Income Tax Schedule

16.9.3 Other Piecewise Linear Functions

You can see from the tax schedule example above that any piecewise linear function can be expressed using various auxiliary variables and Complementarity statements. We give some more examples below.

16.9.3.1 Z = ABS(X)

Here ABS is for the Absolute Value function (see, for example, section 4.4.4 of GPD-2). See Figure 16.9.3.1 on the next page.

Since ABS(X) = MAX(X, –X), it is a simple matter to use the details in section 16.9.1.2 above to express

Z = ABS(X)

in TABLO language as follows:

```
VARIABLE (LEVELS) W ;
EQUATION (LEVELS)  E_W      W = Z – X ;
COMPLEMENTARITY (Variable = W, Lower_Bound = 0)  C_Z
    W + (2*X) ;
```

16.9.3.2 MIN(X,Y) = 0

This relation between X and Y [see Figure 16.9.3.2 on the next page] can be expressed via the statement

```
COMPLEMENTARITY (Variable = X, Lower_Bound = 0)  Comp1   Y ;
```

16.9.3.3 Z = MIN(X,Y) Revisited

If Z = MIN(X,Y)

then 0 = MIN(X – Z, Y – Z) .

Hence section 16.9.3.2 above suggests an alternative to that in section 16.9.1.4 for expressing Z = MIN(X,Y), namely:

```
VARIABLE (LEVELS) U # Difference between X and Z # ;
EQUATION (LEVELS) E_U    U = X – Z ;
COMPLEMENTARITY (Variable = U, Lower_Bound = 0)  C_MIN     Y – Z ;
```

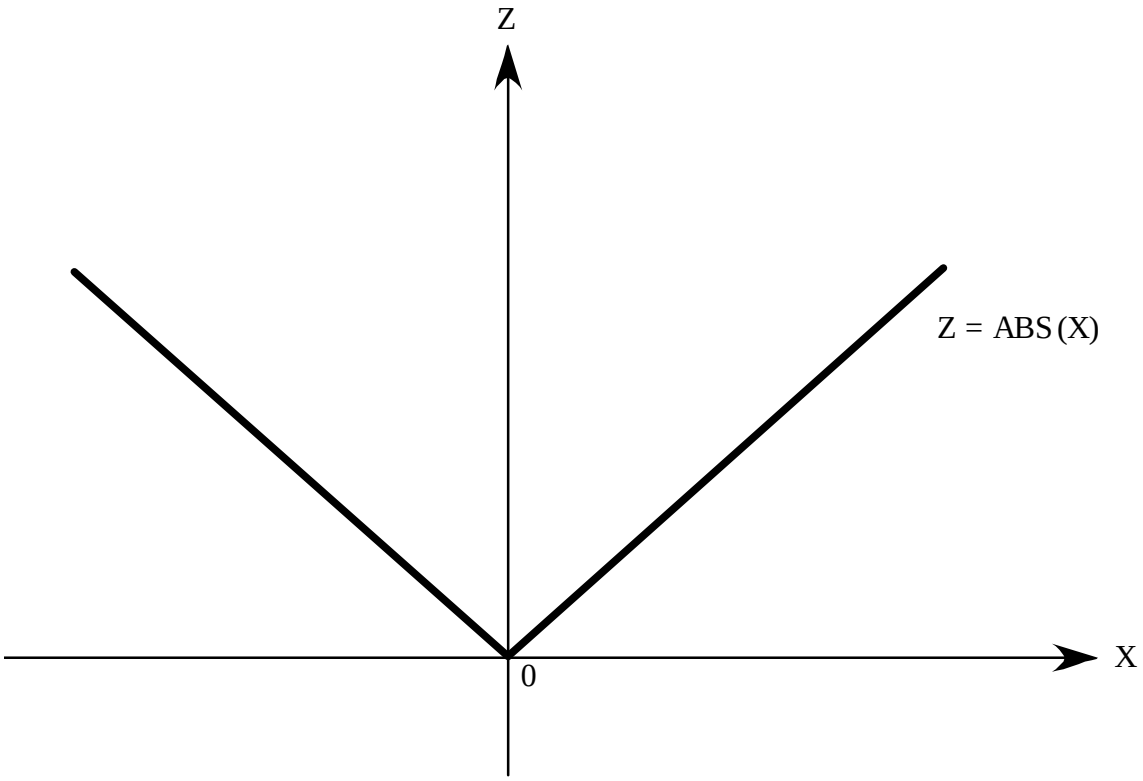


Figure 16.9.3.1 : Z = ABS(X)

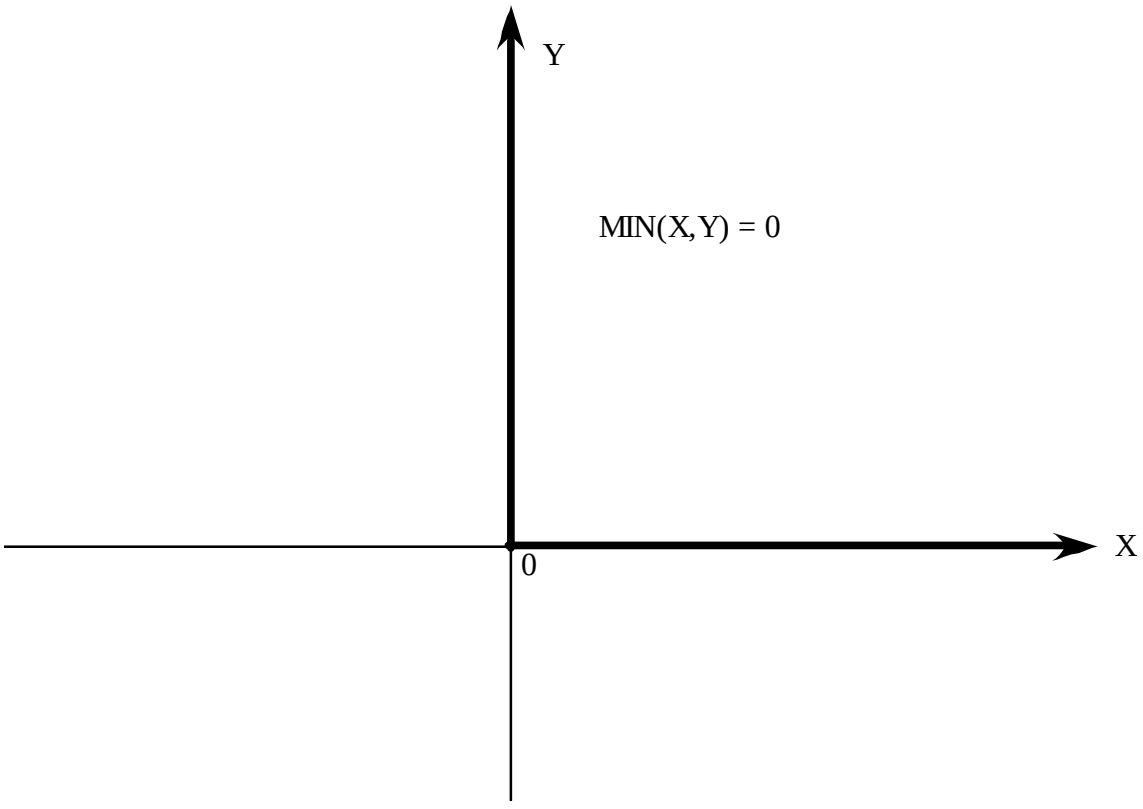


Figure 16.9.3.2 : MIN(X,Y)=0

16.9.3.4 MAX(X,Y) = 0

This relation between X and Y [see Figure 16.9.3.4 on the next page] can be expressed via the statement

```
COMPLEMENTARITY (Variable = X, Upper_Bound = 0)  Comp2   Y ;
```

16.9.3.5 Z = MAX(X,Y) Revisited

If Z = MAX(X,Y)

then 0 = MAX(X – Z, Y – Z) .

Hence section 16.9.3.4 above suggests an alternative to that in section 16.9.1.2 for expressing Z = MAX(X,Y), namely:

```
VARIABLE (LEVELS) U # Difference between X and Z # ;
EQUATION (LEVELS) E_U    U = X – Z ;
COMPLEMENTARITY (Variable = U, Lower_Bound = 0)  C_MAX     Y – Z ;
```

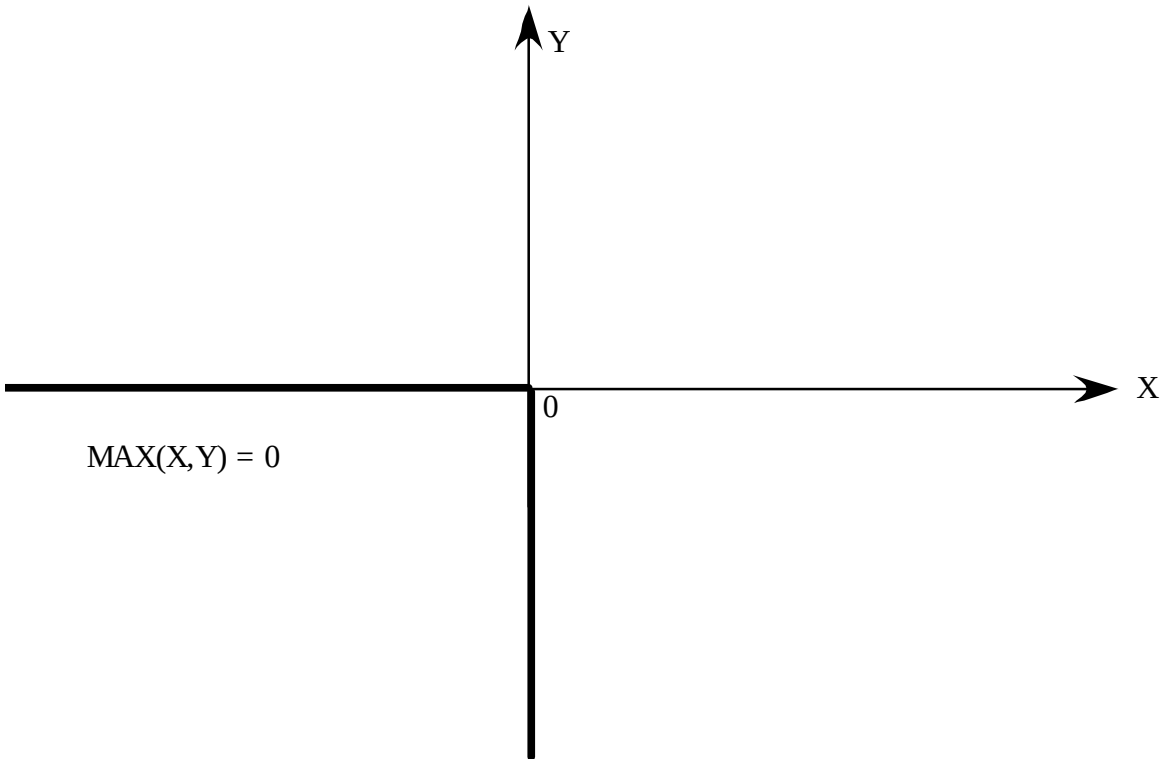


Figure 16.9.3.4 : MAX(X,Y) = 0

16.10 Example: ORANIGRD (Ensuring Investment Stays Non-negative)

In many models, investment is determined endogenously as a function of rates of return. If the rate of return in an industry increases, this is a signal to increase investment in that industry, while if the rate of return falls, investment falls. In the latter case, care must be taken to ensure that the functional relationship between investment and the rate of return does not result in negative investment.

In this example,

- we show how negative investment can occur in a simple modification of the ORANIGRD model.
- we show how to modify the functional relationship between investment and the rate of return using a Complementarity to ensure that investment is always non-negative.

ORANIGRD (see section 1.13 of GPD-8) is a recursive dynamic version of the well-known ORANIG model. [The "RD" at the end of ORANIGRD stands for "recursive dynamic".] The ORANIGRD model was built by Mark Horridge.

In ORANIGRD,

- GROSSRET stands for the GROSS rate of RETurn in an industry – this is equal to capital rentals divided by the value of the capital stock.
- GROSSGRO stands for the GROSS GROwth rate in investment – this is equal to the value of investment divided by the value of the capital stock.

The files for the examples described here can be found in zip file **ORANIGRD.ZIP** supplied with GEMPACK. We are grateful to Mark Horridge who suggested the examples described in this section.

16.10.1 OGRD1.TAB and Negative Investment

ORANIGRD.TAB contains several alternative ways of determining investment.[203]

In OGRD1.TAB we have added an alternative way of determining investment (added to the bottom of the TAB file). This modified way of determining investment is the simple rule

delGGRO = alpha * delGRET (INV5A)

where delGGRO is a linear variable equal to the change in GROSSGRO and delGRET is a linear variable equal to the change in GROSSRET. The simple investment rule INV5A above says that the change in GROSSGRO is alpha times the change in GROSSRET (where alpha is a parameter which is set equal to 3 in OGRD1.TAB).

The problem with this (and other similar) investment rules is that, if GROSSRET falls sufficiently, the resulting negative change in GROSSRET may result in negative GROSSRET and hence in negative investment.

You can see this if you run the simulation in OGRD1.CMF in which a uniform negative technological progress is applied to all primary factors.[204] [The variable **a1prim** is shocked by 2 which indicates technical regress.] Investment in industry "OthMining" [Other Mining] falls by 108 percent, which means that investment has become negative. [To see this, look at the results for variable **x2tot** which is the percentage change in investment by industry.]

16.10.2 OGRD2.TAB – A Complementarity to Ensure Non-negative Investment

In the levels, the simple rule INV5A above corresponds to a linear (straight line) relationship between GROSSGORO and GORSSRET as shown in Figure 16.10.2a. This will result in negative GROSSGRO and hence negative investment if GROSSRET falls below the value indicated by point N in that figure.

The levels form of the equation is

---

[203] To see these, search for "finv" in ORANIGRD.TAB.

[204] First condense the model by running TABLO taking inputs from Stored-input file OGRD1.STI. Then run GEMSIM.

GROSSGRO = alpha*GROSSRET + FINV5

where FINV5 is the value of the Y-intercept (that is, the value of GROSSGRO when GROSSRET=0).[205]

The obvious way to modify the simple investment rule INV5A above is to use it as long as it keeps investment positive but then to override it if it would imply negative investment. For example, use INV5A if it results in positive investment, otherwise set investment equal to zero. This would result in GROSSGRO being determined by the function shown in Figure 16.10.2b. The function there has the equation

GROSSGRO = MAX(alpha*GROSSRET + FINV5 , 0) . (INV5B)

Alternatively, you may prefer to always leave a small positive investment in each industry. This is what we have done in OGRD2.TAB in which the relationship between GROSSGRO and GROSSRET is

GROSSGRO = MAX(alpha*GROSSRET + FINV5 , GGRO_LBOUND) . (INV5C)

where **GGRO_LBOUND** is a lower bound on the possible values for GROSSGRO. We have set GGRO_LBOUND equal to one-tenth of the initial (that is, pre-simulation) value of GROSSGRO in each industry. See Figure 16.10.2c.

This modified investment rule INV5C is a piecewise linear function involving MAX. As we have seen above (section 16.9.1), this can be represented accurately via a Complementarity statement. This is done in OGRD2.TAB where the Complementarity statement is[206]

```
Complementarity (Variable = GROSSGRO_L,
                 Lower_Bound = GGRO_LBOUND)
 C_FINV5     # Simple investment rule - complementarity version #
  (all,i,IND) GROSSGRO_L(i) - [GGRO_GRET(i)*GROSSRET_L(i) + FINV5_L(i)] ;
```

You can see that this keeps investment non-negative if you run the simulation in OGRD2.CMF in which the same uniform negative technological progress is applied to all primary factors as in OGRD1.CMF.[207] Investment in industry "OthMining" [Other Mining] now falls by just 90 percent (since GROSSGRO is not allowed to fall below the lower bound equal to 10% of its original value and since capital stock does not change in this simulation). [To see this, look at the results for variable **x2tot** which is the percentage change in investment by industry.]

Of course there are several other investment rules which ensure that investment does not become negative. Many would involve MAX (as did our example above).[208]

---

[205] Clearly if FINV5 is positive, there will never be negative investment. But negative investment is possible if FINV5 is negative.

[206] What we have called "alpha" is called GGRO_GRET in OGRD2.TAB. GROSSGRO_L is a levels variable which is equal to the coefficient GROSSGRO.

[207] First condense the model by running TABLO taking inputs from Stored-input file OGRD2.STI. Then run GEMSIM.

[208] In the MONASH model [Dixon and Rimmer (2002)], a logistic curve is used. This only ensures non-negative investment if the model is solved very accurately since this logistic curve gets very close to the horizontal axis if rates of return fall sufficiently.

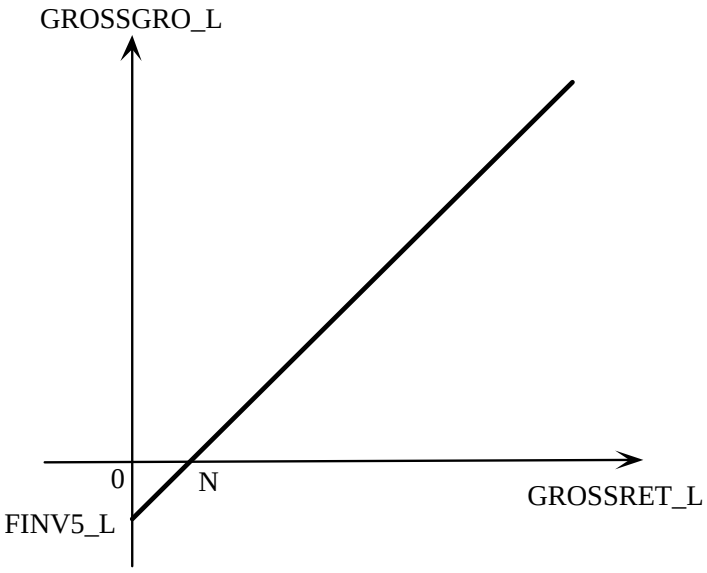


Figure 16.10.2a: Linear Investment Rule INVA

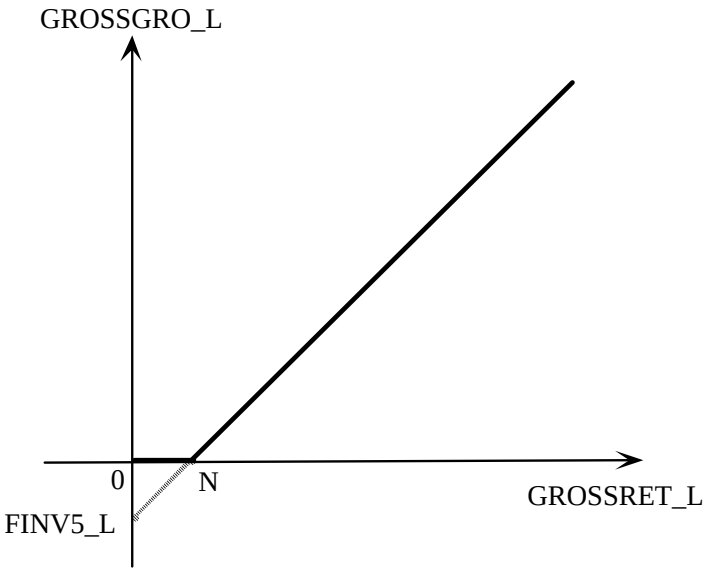


Figure 16.10.2b: Non-negative Investment Rule INVB

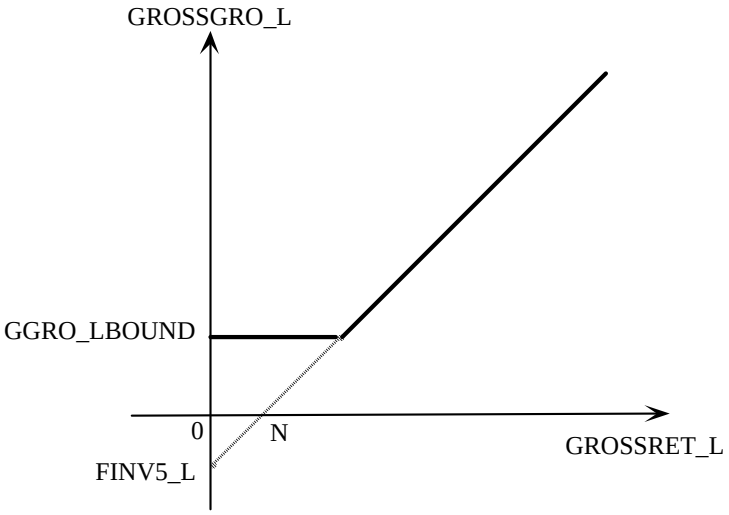


Figure 16.10.2c: Non-negative Investment Rule INVC

CHAPTER 17

17. New Command File Statements

In this chapter we show all the Command file statements which are new for GEMPACK Release 8.0 or were new for Releases 7.0, 6.0 or 5.2.

A complete list of all Command file statements allowed can be found in chapter 18.

17.1 New Command File Statements for Release 8.0

UDC file = yes|NO ; ! Updated Coefficient values - see section 8.5

AVC file = yes|NO ; ! Average Coefficient values - see section 8.5

CVL file = <file name> **;** ! no suffix Coefficient Values – see section 8.6
CVL file = yes ; ! Name is <cmf>.cvl – see section 8.6

CVL system_coefficients = yes ; ! see section 8.7

SLC system_coefficients = yes ; ! see section 8.7

final_level <variable> = ... ; ! see section 5.6

change <variable> = ... ; ! see section 5.7

percent_change <variable> = ... ; ! see section 5.7

structurally singular solve modified equations = yes ;
! see section 15.2.1

complementarity do_approx_run = YES|no ; ! see section 16.6

complementarity steps_approx_run = <number of Euler steps> **;**

complementarity redo_steps = YES|no ;

complementarity redo_step_min_fraction = <number> ;..! eg 0.01

complementarity state/bound_error = FATAL|warn ;

complementarity do_acc_run = no ; ! see section 16.5.6

auto_shocks = YES|no ; ! see footnote section 16.7.2

initialise|initialize coefficients = YES|no|<value> ; ! see section 6.7

17.2 New Command File Statements for Release 7.0

SLC file = YES|no ; ! see section 8.4

xtransfer <header> from file ! see section 4.12.1 of GPD-2
xtransfer unread from file

assertions = YES|no|warn ; ! default is YES
! so assertion checks can be warnings

tshock … ; *! Target shock statement*. For example "tshock xtax = uniform 10 ;". See section 5.5.4

randomize = YES|no ; ! For use with the RANDOM function – see section 4.4.4 of GPD-2

dpn = YES|no ; ! Default is "yes" for Release 7.0 [Previously, default was "no".] See section 4.3.1.

set elements read style = TABLO|flexible ; ! Default is "TABLO" See section 4.4.2.

17.3 New Command File Statements for Release 6.0

<cmf> ! replaced by name of Command file (excluding suffix) in file names– see section 2.5

levels results = YES|no ; ! see section 8.1

minimum subinterval length = * ; ! See section 7.4.5
minimum subinterval fails = STOP|continue ; ! See section 7.4.5

range test updated values = updated|extrapolated|both|no|warn ; ! See section 7.4.5
range test initial values = yes|no|warn ; ! See section 7.4.5

statements to shock variable with none exogenous are ok = yes|NO ;
! See section 5.5.5

scale equations = yes|NO ; ! See section 12.6.6

start with MMNZ|MMNZ1 = <integer value> ;
! Sets initial memory allocation for nonzeros
! in GEMSIM, TABLO-generated programs or SAGEM. See section 13.3.

check-on-read coefficients = yes|warn|NO ; ! See section 4.4 for all of these
check-on-read sets = yes|WARN|no ;
check-on-read elements = YES|warn|no ;
check-on-read all = yes|warn|no ;
check-on-read exact = yes|NO ;

ashock … ; *! Additional shock statement*. For example "ashock xtax = uniform 10 ;". See section 5.5.4

17.4 New Command File Statements for Release 5.2

NAS = yes|NO ; ! should now use "assertions = … ;" instead.
! see footnote in section 6.3

M28 = yes|NO ; ! see section 12.1

SUI = yes|NO ; ! see sections 7.7.2 and 14.3
updated formula initial data = <filename> ; ! see section 7.7.2

NWT = yes|NO ; ! see section 7.5

```
automatic accuracy = yes|NO ;   ! see section 7.4
accuracy figures = <d> ;      ! see section 7.4
accuracy percent = <pp> ;       ! see section 7.4
accuracy criterion = DATA|solution|both ;   ! see section 7.4

SSL = all | last | NONE ;  ! see section 7.7.1
SUP = all | last | NONE ;   ! see section 7.7.1

exogenous|endogenous <variable> = zero|nonzero|positive|negative
        value on file <file> ;  ! see section 5.2.5
! EXAMPLE.
!     exogenous x1 = nonzero value on file DAT1.DAT ;
!     endogenous x1 = zero value on file DAT1.DAT ;


xset...     ! see section 6.6
xsubset
xfile
xwrite
xdisplay


SHOCK RELATED
! For TABLO-generated programs and GEMSIM and SAGEM
! if no shock statements are given, an error occurs.
```

CHAPTER 18

18. Summary of Command File Statements

This contains complete and self-contained documentation of all Command file statements for Release 8.0 of GEMPACK.

Command files were introduced in chapter 2 and also in GPD-1 in sections 2.8, 2.8.1 (Figure 2.8.1) and 2.12.4 (Figure 2.12.4). Command files are used to run simulations with GEMSIM, TABLO-generated programs or SAGEM. In this chapter we give a complete list of the commands which can be used in GEMPACK Command files for running GEMSIM or TABLO-generated programs (see section 18.1 below) and SAGEM (see section 18.2 below).

In this chapter, the actual commands are shown in bold; the surrounding text is commentary.

18.1 Command Files for GEMSIM and TABLO-generated Programs

The detailed rules for preparing Command files for running GEMSIM and TABLO-generated programs are set out below. In the commands listed below, note the following.

1. "|" indicates alternatives. For example,

   'method = johansen | euler | midpoint | gragg ;'

   means that any of 'method = johansen ;', 'method = euler ;', 'method = midpoint ;' or 'method = gragg ;' is accepted.
2. < > shows user-selected text.
3. **[ ]** indicates optional user-selected text.
4. all other text is required (but may be abbreviated as explained under "General Points" below).
5. If a default is specified, this is the value if no such command is given.
6. The special string **<cmf>** is replaced by name of Command file (excluding suffix) in file names provided that the Command file used has suffix .CMF (any case). See section 2.5 for more details.

18.1.1 Command File Statements

METHOD AND STEPS

method = johansen|euler|midpoint|GRAGG ; ! gragg is the default
! EXAMPLE. method = gragg ;
! See section 7.1.2

steps = <list_of_step_numbers> ; ! See section 7.1.2
! There is no default unless you are using automatic accuracy when
! the default is 2,4,6 – see section 7.4.
! EXAMPLE. steps = 2 4 6 ;
! (There is no longer automatic doubling of step numbers, since this is unnecessary.)
! If you specify just one step number, a single multi-step simulation is done.
! If you specify two or three, extrapolation is done also.

subintervals = <number_of_subintervals> ; ! Default is 1.
! EXAMPLE. subintervals = 10 ;
! See section 7.3

CHECKING SET AND ELEMENT LABELLING WHEN READING DATA FROM HA FILES

See section 4.4 for details about these check on read statements..

check-on-read coefficients = yes|warn|NO ;
check-on-read sets = yes|WARN|no ;
check-on-read elements = YES|warn|no ;
check-on-read all = yes|warn|no ;
check-on-read exact = yes|NO ;

CHECKING ELEMENT NAMES WHEN READING SETS FROM HA FILES

set elements read style = TABLO|flexible ; ! Default is "TABLO" see section 4.4.2

AUTOMATIC ACCURACY

automatic accuracy = yes|NO ; ! NO is the default
! use "yes" to specify the accuracy
! see section 7.4 for details

accuracy figures = <d> ; ! 4 is the default (see section 7.4)
! EXAMPLE. accuracy figures = 5 ;

accuracy percent = <pp> ; ! 80 percent is the default (see section 7.4)
! EXAMPLE. accuracy figures = 90 ;

accuracy criterion = DATA|solution|both ; ! data is the default (see section 7.4)
! EXAMPLE.. accuracy criterion = both ;

minimum subinterval length = * ; ! default is 0.000001 (1.0e-6)
! EXAMPLE. minimum subinterval length = 0.000000001 ;
! EXAMPLE. minimum subinterval length = 1.0e-9 ;
! exponential notation 1.0e-9 is allowed
! This specifies the minimum subinterval length allowed.
! See section 7.4.5

minimum subinterval fails = STOP|continue ;
! This tells what happens if a subinterval has the minimum length and
! accuracy criterion or a range test fails. Either STOP (the default)
! or continue. In the "continue" case, the software continues
! with the minimum subinterval length, even if range tests or
! accuracy criterion fails.
! See section 7.4.5

range test updated values = updated|extrapolated|BOTH|no|warn ;
! default is "both"
! This tells which updated values have range restrictions (if there are
! any in the model) tested.
! "updated" only tests values when they have been updated (but not after extrapolation).
! "extrapolated" only tests values immediately after extrapolation (but not
! after updating).
! "both" tests values at both times (updated and extrapolated)
! "no" turns off both sorts of tests. See section 7.4.5

range test initial values = YES|no|warn ; ! default is "yes"
! This tells whether initial values have range restrictions (if there are
! any in the model) tested.
! "yes" means they are tested, "no" means they are not tested.
! See section 7.4.5

DATA FILES

file <logical_name> = <actual_name> ;

! EXAMPLE. file iodata = sj.dat ; ! include suffix
! for original data files or for new (WRITE) files
! This is not necessary if the actual file name is given in the TABLO Input file
! see section 4.1

updated file <logical_name> = <actual_name> ; ! for final updated data

! EXAMPLE. updated file iodata = sjlb.upd ;
! include suffix
! see section 4.2

updated terminal data = <file_name> ;

! for the updated version of data read initially from the terminal (see section 4.2.2)
! EXAMPLE. updated terminal data = sjlb_term.upd ;

intermediate file <logical_name> = <actual_name> ;

! For intermediate updated versions. See section 4.5.
! NO SUFFIX in <actual_name>; '.UD3', '.UD4' etc will be added.
! On most machines, if you use a suffix of length 3 (such as '.UPD')
! for the updated version, the default is to use the name minus this suffix.
! Then no 'intermediate file' commands need to be included.
! See section 4.5.1 for the default if an "intermediate file"
! statement is not included for any logical file.
! EXAMPLE. intermediate file iodata = sjlb ;

intermediate extra data = <actual_name> ;

! For intermediate updated versions of data read initially from the terminal or
! from text files, or whose initial values are assigned via FORMULA(INITIAL) statements.
! NO SUFFIX. As above, suffixes '.UD3' etc will be added.
! Default <actual_name> is "GPXXX" - see section 4.5.1.
! EXAMPLE. intermediate extra data = sjlb_term ;

! NOTE. Often no 'intermediate file ...' or 'intermediate extra ...' commands are
! necessary since the defaults will usually suffice. (See section 4.5.1 for
! advice on choosing names for updated data files so that defaults for intermediate
! updated file names give valid file names and for information as to
! when an 'intermediate extra' file is needed.)

EQUATIONS AND BCV FILES

The following was new for **Release 6.0** GEMPACK. The default is that GEMSIM and TABLO-generated programs do not save an Equations file or a BCV file
so it is **no longer necessary to have a statement *equations file = <file_name> ;***
See section 9.1 for details.

Equations file = <file_name> ; ! To create and save a new Equations file
! EXAMPLE. Equations file = sj ;
! omit suffix '.eq4'

! In Release 7.0 or later, the Equations file is never created
! unless you use a Command file and include this statement.
! See section 9.1 for details.
! For information about saving an Equations file in earlier
! releases, see the footnote in section 9.1.

model = <name> ; ! to specify the model name in a new Equations file
! EXAMPLE. model = sj ;
! If this statement is omitted, *model =< blanks> ;* is assumed
! see section 9.1.3

version = <integer> ; ! to specify the version number on a new Equations file.
! DEFAULT is 1 (see section 9.1.3).
! EXAMPLE. version = 3 ;

identifier = <identifier> ; ! to specify the model identifier on a new Equations file
! DEFAULT is all blanks (see section 9.1.3)
! EXAMPLE. identifier = Stylized Johansen, standard data ;

use Equations|BCV file <file_name> ; ! To use existing Equations and BCV files
! If you specify an Equations file to use, the default is to use a BCV file with the same
! name (unless you have a separate 'use BCV file' command)
! EXAMPLE. use Equations file sj ;
! no suffix
! see section 9.2.1

BCV file = <file_name> ;
! To specify the name of the (new) BCV (Base Coefficient Values) file to be created.
! EXAMPLE. BCV file = sj ;
! omit suffix '.bcv'

! In Release 7.0 or later, a BCV file is never created
! unless you use a Command file and include this statement.
! See section 9.2 for details.
! For information about saving a BCV file in earlier
! releases, see the footnote in section 9.2.

! *equations file = <file_name> ;* does not automatically
! save a BCV file as well, you need a *BCV file = <file_name> ;* as well
! EXAMPLE. Equations file = sj ;
! BCV file = sj ;
! see section 9.2 for details

OTHER FILES

Solution file = <file_name> ;

! EXAMPLE. Solution file = sjlb ;
! omit suffix '.sl4'
! see section 8.1
! default if statement is not present *solution file = command file stem ;*
! provided the Command file ends with ".CMF" (any case allowed)

display file = <file_name> ; ! include suffix

! EXAMPLE. display file = sj.dis ;
! default if statement is not present *display file = command file stem + .dis ;*
! provided the Command file ends with ".CMF" (any case allowed)
! see section 2.5.3 for details

Extrapolation Accuracy file = yes|NO ; ! default is NO

! Use 'yes' to get an XAC file
! see section 7.2.3
! An XAC file is not created unless you are extrapolating
! from two or three multi-step solutions.
! EXAMPLE. extrap acc file = yes ;

Auxiliary files = <file_name> ;

! (a) GEMSIM (see section 3.1)
! This statement is mandatory.
! It specifies the names of the GEMSIM Statement and Table files.

! EXAMPLE. Auxiliary files = sj ; ! no suffix

! (b) TABLO-generated programs (see section 3.2)
! This is for name of Auxiliary Statement and Table files.
! Your TABLO-generated program contains a default name which
! is usually ok. Only use this if you need to change it
! (perhaps to indicate the directory these are in).
! This corresponds to option NAX (see chapter 14).

! EXAMPLE. Auxiliary files = c:\sj\sjln ; ! no suffix

Submatrix Data file = <file_name> ; ! No longer supported in Release 5.2 or later

Set-up file = <file_name> ; ! No longer supported in Release 5.2 or later

TABLO-LIKE STATEMENTS IN COMMAND FILES

! These statements are described in section 6.6

xset....

xsubset...

xfile....

xwrite...

xdisplay.....

xtransfer <header> from file

xtransfer unread from file

! You can put these TABLO-like statements in command files and they will be carried
! out as if they were appended to the end of the original TABLO Input file.

! Syntax and semantics similar to SET, SUBSET, FILE, WRITE, DISPLAY, TRANSFER statements
! in TABLO.

```
! EXAMPLE.    xfile(new,text) sjout ;
!             xwrite alphafac to file sjout ;
!             file sjout = SJLBOUT.DAT ;
```

CLOSURE RELATED

! The use of these commands is discussed in section 5.2

save Environment|LU file <file_name> ;

! EXAMPLE. save Environment file sjxfac ;
! omit suffix '.en4'
! see sections 5.2.1 and 9.3

use Environment|LU file <file_name> ;

! EXAMPLE. use Environment file sjxfac ;
! omit suffix '.en4'
! see sections 5.2.2 and 9.3
! If 'use Environment file', just the closure is read from it.

! If 'use LU File', the closure AND LU decomposition are read from it – see section 9.3.
! You cannot 'use LU file' unless you are also using existing Equations and BCV files.

take closure from Environment|LU|Solution file <file_name> ;

! EXAMPLE. take closure from Environment file sjxfac ;
! omit suffix '.en4'
! see sections 5.2.2 and 9.3

! Note that 'take closure from Environment file' has exactly the
! same effect as 'use Environment file' – see section 5.2.2.

! If 'take closure from LU File', the closure is read from it, but the LU decomposition is
! calculated from scratch – see section 9.3.

modify closure from Environment|LU|Solution file <file_name> ;

! The closure is read from this file. Then you use 'swap',
! 'exogenous' and/or 'endogenous' commands (see below) to
! modify this closure.
! see section 5.2.3

! If 'modify closure from LU file', the LU decomposition on the file is not used (see section 9.3).

swap <v1 [component_list]> = <v2 [component_list]> ;

! v1 v2 are variables
! EXAMPLES. swap v1 1-3 = v2 6-8 ;
! swap p_XF("labor","s1") = cR ;
! see section 5.2

exogenous|endogenous <v1 [component_list]> <v2 [component_list]> ... ;
rest exogenous|endogenous ;

! EXAMPLE. exogenous xfac p_XF("labor",SECT) facind 3-15 phi ;
! rest endogenous ;
! See section 5.2.1

exogenous | endogenous <variable> = zero | nonzero | positive | negative
value on file <file> ;

! see section 5.2.5
! EXAMPLE. exogenous x1 = nonzero value on file DAT1.DAT ;
! endogenous x1 = zero value on file DAT1.DAT ;

HARWELL PARAMETER

Harwell parameter = <u_value> ; ! default is 0.1

! EXAMPLE. Harwell parameter = 0.4 ;
! see section 12.1

VERBAL DESCRIPTION

verbal description =
<line 1>
<line 2> etc, until
<last line> ;

! last line ENDS with **;**
! (There can be other ';' earlier, as long as they are not at the end of their line.)
! Each line must contain no more than 79 characters.
! see section 8.1
! EXAMPLE. verbal description =
Stylized Johansen; standard data.
Labor shock. ;

! The method and steps and also the Command file name are added
! by the program to the Verbal description.

SHOCK RELATED

! The use of these commands is discussed in section 5.5.1.

! Before Release 5.2, the default was that all exogenous variables are shocked with
! all shocks equal to 1.0

! For Release 5.2 and later this was changed to the following:
! For TABLO-generated programs and GEMSIM (and SAGEM)
! if no shock statements are given, an error occurs.

shock <v1 [component_list]> = file <file_name> ;
! read shocks from this file
! EXAMPLE. shock v1 = file v1.shk ;

shock <v1 [component_list]> = uniform <value> ;
! all components (in list) given the same shock
! EXAMPLE. shock v1 = uniform 0.3 ;

shock <v1 [component_list]> = <list_of_values> ;
! If present, the component numbers in component_list
! must be in increasing order.
! EXAMPLES. shock v1 2 4 6 = 0.2 0.4 -0.1 ;
! shock p_XF("labor",SECT) = 0.2 0.4 ;

shock <v1 [component_list]> = select from file <file_name> ;

! Read shocks from this file and select
! values corresponding to components listed
! EXAMPLE. shock v1(IND) = select from file v1.shk ;

shock <v1 [component_list]> = select from <list_of_values> ;

! Select shock values corresponding to components listed
! (If present, the component numbers in
! component_list must be in increasing order.)

! EXAMPLE. shock v1 2 4 = select from 0.1 0.2 0.4 -0.1 ;
! This shocks components 2 and 4 by 0.2 and -0.1 respectively.

! You need one 'shock' command for each variable all
! or some of whose components are shocked.

statements to shock variable with none exogenous are ok = yes|NO ;
! The default is "no" which means (as before) that a statement shock <variable> = ... ;
! is only allowed if <variable> has at least one exogenous component.
! See section 5.5.5.

`ashock … ;` *! Additional shock statement.* For example "ashock xtax = uniform 10 ;".
`tshock … ;` *! Target shock statement.* For example "tshock xtax = uniform 10 ;".
!See section 5.5.4

! Shocks where levels value is known – new for Release 8.0
`final_level <variable> = ... ;` ! see section 5.6
`change <variable> = ... ;` ! see section 5.7
`percent_change <variable> = ... ;` ! see section 5.7

CUMULATIVELY-RETAINED ROWS

! The use of these commands is discussed in section 8.1.

cumulatively-retained endogenous <list> ;

! default is: all endogenous variables (including any variables backsolved for)

where the things in <list> can be any of

%all ! this means all (endogenous)

%macro ! this means macros [i.e. variables with 1 component]

! (which are endogenous)

%scalar ! this means the same as '%macro'

<v1 [component_list]> <v2 [component_list]> ...

<list_of_set_types> such as

(COM) (COM,IND) ...

! EXAMPLE.

! cum endog p_XFAC p_XF("labor",SECT) %macro (SECT,FACT) pcom 1-2 ;

VARIABLES ON EXTRAPOLATION ACCURACY FILE

XAC-retained <list> ; ! default is all endogenous variables

! (including any variables backsolved for)

! Things in <list> as for 'cumulatively-retained endogenous'

! Affects variables on Extrapolation Accuracy file (if one requested)

! Can also affect automatic accuracy

! See section 7.2.4.2

SUBTOTALS

subtotal <v1 [component_list]> <v2 [component_list]> ... =
<description> ;

! The shocks to sum over to produce the subtotal are those given to
! all the variables and components listed before the '=' sign.
! <description> is limited to 77 characters.
! see section 11.1

! EXAMPLE. subtotal px 1-10 15-20 pm 2-4 zz = subtotal 2 ;

COMPLEMENTARITY

! Complementarities were new for Release 8.0

! see section 16.6 for all of these

complementarity do_approx_run = YES|no ;

complementarity steps_approx_run = <number of Euler steps> **;**

complementarity redo_steps = YES|no ;

complementarity redo_step_min_fraction = <number> ;..! eg 0.01

complementarity state/bound_error = FATAL|warn ;

auto_shocks = YES|no ; ! see footnote section 16.7.2

GEMSIM AND TABLO-GENERATED PROGRAM OPTIONS

log file|only = <file_name> ; ! default is no log file
! EXAMPLE. log file = sjlb.log ! include suffix
! 'log file = ... ;' is used to direct output to both the terminal and a Log file.
! 'log only = ... ;' is used to direct output to just the Log file (not to the terminal).
! For more details, see section 5.3.3 in GPD-1.

log file|only = yes ;
! This is equivalent to *log file|only = command file stem + .log ;*
! provided your Command file ends in the suffix ".CMF" (any case is allowed)
! see section 2.5.2 for details
! EXAMPLE. log file = yes ;
! If your Command file is *sjlb.cmf* then log file is *sjlb.log*

simulation = YES|no ; ! default is YES.
! 'simulation = no ; is used to just create an Equations file, for example.
! This corresponds to option NSM - see section 14.3.

levels results = YES|no ; ! default is YES
! If the levels results are available, they will be written on the Solution file
! Use 'levels results = no ;' if you do not want these solution results
! see section 8.1 for details.

SLC file = YES|no ; ! default is YES
! write all pre-simulation Coefficient values to the SLC file
! new for Release 7.0 for use with AnalyseGE program
! see section 8.4

! UDC, AVC, CVL files new for Release 8.0 for use with AnalyseGE program
UDC file = yes|NO ; ! Updated Coefficient values
! see section 8.5

AVC file = yes|NO ; ! Average Coefficient values - see section 8.5

CVL file = <file name> **;** ! no suffix Coefficient Values – see section 8.6
CVL file = yes ; ! Name is <cmf>.cvl – see section 8.6

CVL system_coefficients = yes ; ! see section 8.7

SLC system_coefficients = yes ; ! see section 8.7

! Initialising values of Coefficients – new for Release 8.0

initialise|initialize coefficients = YES|no|<value> ; ! see section 6.7

! Debugging option – new for Release 8.0

structurally singular solve modified equations = yes ; ! see section 15.2.1

OPTIONS (continued)

! The commands below correspond to some of the options offered at the start of
! TABLO-generated programs or the "Further Options" screen for GEMSIM.
! (Unless stated otherwise, these options are documented in chapter 14.)

CPU = yes|NO ; ! default is NO.
! Use 'CPU = yes ;' to get CPU times reported.
! (This may not produce meaningful results on some machines
! see section 14.6

EAA = yes|no ;
! See section 14.6 to see what "echoing activity" means.
! (a) If you are carrying out a simulation, the default is NO
! which means activity is only echoed during the first step.
! In this case, use "eaa = yes ;" to have activity echoed
! during all steps of a multi-step calculation.
! (b) If you are not carrying out a simulation, the default is YES
! which means all activity is echoed. In this case,
! use "eaa = no ;" to have no activity echoing.

NRP = yes|NO ; ! default is NO.
! Use 'NRP = yes ;' to not reuse pivots.

NSC = YES|no ; ! No longer supported – see sections 14.4 and 9.1

NSE = YES|no ; ! No longer supported – see sections 14.4 and 9.2

NEQ = yes|NO ; ! default is NO.
! Use 'NEQ = yes ;' to do no equations, only formulas etc
! (hence no simulation).

NDS = yes|NO ; ! default is NO.
! Use 'NDS = yes ;' to do no displays.

NWR = yes|NO ; ! default is NO.
! Use 'NWR = yes ;' to do no writes.

NUD = yes|NO ; ! default is NO.
! Use 'NUD = yes ;' to do no FINAL updates.
! Intermediate ones will still be done.

NAS = yes|NO ; ! default is NO. See footnote in section 6.3
! Use 'NAS = yes ;' to suppress checking of assertions.
! Now prefer to use "assertions = … ;" statement – see below.

assertions = YES|no|warn ; ! default is YES
! New for Release 7.0
! so assertion checks can be warnings

NIR = yes|NO ; ! default is NO. See section 12.1.
! Use 'NIR = yes ;' to turn off iterative refinement of solutions.

IZ1 = YES|no ; ! default is YES.[209]
! Use 'IZ1 = no ;' to keep coefficients which are zero at step 1.
! See section 12.5 for details.

KZ2 = yes|NO ; ! default is NO.
! Use 'KZ2 = yes ;' to keep coefficients which are zero at steps 2,3,...

[209] In Release 5.0, the default was NO.

NWE = yes|NO ; ! default is NO.
! Use 'NWE = yes ;' to suppress any warnings about equations not
! being solved accurately.

RQF = <integer> ; ! to change the default number of figures agreement
! required for '"machine accuracy" on Extrapolation Accuracy files.

M28 = yes|NO ; ! default is NO which means use the new MA48 Subroutines
! Use 'M28=yes ;' to use the MA28 Harwell subroutines
! for solving sparse linear equations
! see section 12.1 for details

SUI = yes|NO ;
updated formula initial data = <filename> ;
!default is NO
! Use 'SUI=yes ;' to save the updated values of coefficients
! whose initial values are set via FORMULA(INITIAL)
! Updated values are written to the file <filename>
! See section 14.3 above for details
! EXAMPLE. sui=yes ;
! updated formula initial data = sjlbfi.upd ;

NWT = yes|NO ; ! default is NO
! Use 'NWT=yes ;' to use the Newton method
! to solve the equations - see section 7.5 above.

SSL= all | last | NONE ; ! default is NONE
! SSL = all ; is used to save files from all separate multi-step
! solutions as well as the extrapolated solution
! SSL= last ; saves just the last of the multistep solutions
! see section 7.7.1 for details

SUP = all | last | NONE ; ! default is NONE
! 'SUP = all ;' saves the updated data files from all multisteps
! 'SUP = last ;' to save the updated data files from the last multistep
! see section 7.7.1 for details

scale equations = yes|NO ; ! default is NO
! Option new for Release 6.0
! See section 12.6.6 for details

start with MMNZ|MMNZ1 = <integer value> ; ! Sets initial memory allocation for
! nonzeros in GEMSIM or TABLO-generated programs. See section 13.3.
! [Also applicable to SAGEM.]

randomize = YES|no ; ! For use with the RANDOM function – see section 4.4.4 of GPD-2

! (see section 4.3 and 14.5 for details about the options below)

DWS = yes|NO ; ! default is NO.
! Use 'DWS = yes ;' to do terminal writes and displays at all steps.

TWC = yes|NO ; ! default is NO which means writes to the terminal
! will be done in row order.
! Use 'TWC = yes ;' to do such writes in column order.

display width|length|decimals = <integer> ;
! to change page width or length, or number of figures after
! decimal point, in display files.
! Width must be between 70 and 200,
! Length at least 20 and
! Decimals between 0 and 10.

! (These correspond to options DPW,DPL,DDC - see
! section 14.5.)

DPN = YES|no ; ! default is YES. [Before Release 7.0, default was "no".]
! Use 'DPN = no ;' to ensure that each new coefficient displayed
! starts on a new page.

D1C = yes|NO ; ! default is NO, which means that a 1-dimensional
! array is displayed as a row (across the page).
! Use 'D1C = yes ;' to have each such array displayed
! as a column (down the page).

DOI = yes|NO ; ! default is NO.
! Use 'DOI = yes ;' to suppress "identical" row
! messages on display files.

NEL = yes|NO ; ! default is NO (see section 14.5).
! Use 'NEL = yes ;' to suppress element name/number
! labels on row_order or col_order text files.

FOR DEBUGGING

! You will not need to use these unless the GEMPACK developers ask you to include them to
! give more information about a problem you are encountering.

duplicate headers = DELETE|keep|warn|ignore ;

! *delete* and *keep* mean duplicate headers found anywhere is a fatal error.
! *warn* means only warn about such. *ignore* means ignore them entirely.

! if find duplicate headers on new file, *delete* means delete this file, *keep* means keep it.
!At present, *ignore* is not implemented. At present it really has the same effect as *warn*.

debug options = <number1> <number2> ... ; ! list of debug options

GENERAL POINTS

See section 2.7.

18.1.2 Complete Command File Example for GEMSIM and TABLO-generated Programs

```
!_____________________________________________________________________
!
! The following GEMPACK Command file will run the TABLO-generated
!  program for the Stylized Johansen model to carry out a
!  multi-step simulation.
!
! Solution method information
method = gragg ; ! could be omitted as this is the default
steps = 2 4 6 ;


! Data files
file iodata = sj.dat ;
updated file iodata = <cmf>.upd ;


! Simulation part
solut file = sjlbg ;  ! Usually omitted – see section 2.5.1
use Environment file sjxfac ;
shock p_xfac 1 = 10 ;
verbal description =
Increase labor by 10 per cent.
Standard closure. ;


! If want Equations file (not necessary or the default)
equat file = sj ;  ! creates a new Equations file
 model = sj ;  !
 version = 1 ; ! (Could be omitted as this is the default)
 identifier = Stylized Johansen. Standard data ;


! Options (just examples)
log file = yes ; ! Log file name taken from Command file
                 !    name (see section 2.5
           ! output goes to log file and also to terminal
CPU = yes ;           ! report CPU times
extrapolation accuracy file = yes ;
!_____________________________________________________________
```

18.2 Command Files for Running SAGEM

The commands in Command files for running SAGEM are similar to the relevant ones (dealing with closure, shocks etc) for GEMSIM and TABLO-generated programs. The details are given below.

LIST OF COMMANDS ACCEPTED ON SAGEM COMMAND FILES

use Equations file <file_name> ; ! MANDATORY (see section 9.1.1)
! EXAMPLE. use Equations file sj ;
! omit suffix '.eq4'

Solution file = <file_name> ;
! EXAMPLE. Solution file = sjlb ;
! omit suffix '.sl4'
! The following is new for Release 6.0 GEMPACK
! default if statement is not present *solution file = command file stem ;*
! provided the Command file ends with ".CMF" (any case allowed)
! see section 2.5 for details

Harwell parameter = <u_value> ; ! default is 0.1
! EXAMPLE. Harwell parameter = 0.4 ;

VERBAL DESCRIPTION (MANDATORY)

verbal description =
<line 1>
<line 2> etc, until
<last line> ; ! last line ENDS with ;

! (There can be other ';' earlier, as long as they are not at the end of their line.)
! Each line must contain no more than 79 characters.
! See section 8.1.

! EXAMPLE. verbal description =
Stylized Johansen; standard data.
Labor shock. ;
! The Command file name is added by SAGEM to the Verbal description.

CLOSURE RELATED

! The use of these commands is discussed in section 5.2.

save Environment|LU file <file_name> ;

! EXAMPLE. save Environment file sjxfac ;
! omit suffix '.en4'
! see sections 5.2.1 and 9.3

use Environment|LU file <file_name> ;

! EXAMPLE. use Environment file sjxfac ;
! omit suffix '.en4'
! see sections 5.2.2 and 9.3
! If 'use Environment file', just the closure is read from it.

! If 'use LU File', the closure AND LU decomposition are read from it – see section 9.3.

take closure from Environment|LU|Solution file <file_name> ;

! EXAMPLE. take closure from Environment file sjxfac ;
! omit suffix '.en4'
! see sections 5.2.2 and 9.3

! Note that 'take closure from Environment file' has exactly the
! same effect as 'use Environment file' – see section 5.2.2.

! If 'take closure from LU File', the closure is read from it, but the LU decomposition is
! calculated from scratch – see section 9.3.

modify closure from Environment|LU|Solution file <file_name> ;

! The closure is read from this file. Then you use 'swap',
! 'exogenous' and/or 'endogenous' commands (see below) to
! modify this closure.
! see section 5.2.3

! If 'modify closure from LU file', the LU decomposition on the file is not used (see section 9.3).

swap <v1 [component_list]> = <v2 [component_list]> ;
! v1 v2 are variables

! EXAMPLES. swap v1 1-3 = v2 6-8 ;
swap p_XF("labor","s1") = p_cR ;

exogenous | endogenous <v1 [component_list]> <v2 [component_list]> ... ;
rest exogenous | endogenous ;

! EXAMPLE. exogenous xfac pcom facind 3-15 phi ;
! rest endogenous ;

exogenous | endogenous <variable> = zero | nonzero | positive | negative
value on file <file> ;

! see section 5.2.5
! EXAMPLE. exogenous x1 = nonzero value on file DAT1.DAT ;
! endogenous x1 = zero value on file DAT1.DAT ;

SHOCK RELATED

! The use of these commands is discussed in section 5.5.

! You must have at least one shock statement or an error occurs.

shock <v1 [component_list]> = file <file_name> ;
! read shocks from this file
! EXAMPLE. shock v1 = file v1.shk ;

shock <v1 [component_list]> = uniform <value> ;
! all components (in list) given the same shock
! EXAMPLE. shock v1 = uniform 0.3 ;

shock <v1 [component_list]> = <list_of_values> ;
! If present, the component numbers in
! component_list must be in increasing order.

! EXAMPLE. shock v1 2 4 6 = 0.2 0.4 -0.1 ;

shock <v1 [component_list]> = select from file <file_name> ;
! Read shocks from this file and select
! values corresponding to components listed

! EXAMPLE. shock v1(IND) = select from file v1.shk ;

shock <v1 [component_list]> = select from <list_of_values> ;
! Select shock values corresponding to components listed
! (If present, the component numbers in component_list must be
! in increasing order.)

! EXAMPLE. shock v1 2 4 = select from 0.1 0.2 0.4 -0.1 ;
! This shocks components 2 and 4 by 0.2 and -0.1 respectively.

! You need one 'shock' command for each variable
! all or some of whose components are shocked.

INDIVIDUALLY-RETAINED AND CUMULATIVELY-RETAINED ROWS/COLUMNS

! The use of these commands is discussed in chapter 10.

individually-retained exogenous|endogenous <list> ;
! default for individually-retained exogenous is: none
! default for individually-retained endogenous is: all endogenous variables
! See section 10.1.1.

cumulatively-retained endogenous <list> ;
! default is: all endogenous variables
! See section 10.1.2.

where (in all cases) the things in <list> can be any of

%all ! this means all of the appropriate set (shocked or endogenous)

%none ! this means none of the appropriate set

%macro ! this means macros [i.e. variables with 1 component]
! in the appropriate set)

%scalar ! this means the same as '%macro'

<v1 [component_list]> <v2 [component_list]>

<list_of_set_types> such as

(COM) (COM,IND) ...

! EXAMPLE. cum endog p_XFAC %macro (SECT,FACT) pcom 1-2 ;

SUBTOTALS

subtotal <v1 [component_list]> <v2 [component_list]> ... =
<description> ;

! The shocks to sum over to produce the subtotal are those given to
! all the variables and components listed before the '=' sign.
! <description> is limited to 77 characters.
! See section 10.2.

! EXAMPLE. subtotal fe 1-10 15-20 pm = subtotal 2 ;

SAGEM OPTIONS

! These correspond to some of the options offered at the start of SAGEM.

log file|only = <file_name> ; ! default is no log file
! EXAMPLE. log file = sjlb.log
! include suffix

! Use 'log file = ... ;' to direct output to both the
! terminal and a Log file.

! Use 'log only = ... ;' to direct output to just the
! Log file (not to the terminal).

! For more details, see section 5.3.3 of GPD-1.

simulation = YES|no ; ! default is YES.
! Use 'simulation = no ;' to just specify closure.
! (This corresponds to option NSM)
! see section 6.1.7

start with MMNZ|MMNZ1 = <integer value> ;
! Sets initial memory allocation for
! nonzeros in SAGEM. See section 13.3.

CPU = yes|NO ; ! default is NO (see section 14.7.1).
! Use 'CPU = yes ;' to get CPU times reported.
! (This may not produce meaningful results on some machines)

NIR = yes|NO ; ! default is NO (see section 12.1).
! Use 'NIR = yes ;' to turn off iterative refinement of solutions.

KZC = yes|NO ; ! default is NO (see section 14.7.1).
! Use 'KZC = yes ;' to have SAGEM keep coefficients which are zero.

M28 = yes|NO ; ! default is NO which means use the new MA48 Subroutines
! Use 'M28=yes ;' to use the MA28 Harwell subroutines
! for solving sparse linear equations
! see section 12.1 for details

NWE = yes|NO ; ! default is NO (see section 14.7.1).
! Use 'NWE = yes ;' to suppress any warnings about equations
! not being solved accurately.

scale equations = yes|NO ; ! default is NO
! See section 12.6.6 for details

GENERAL POINTS

See section 2.7. [These are just as for Command files for GEMSIM and TABLO-generated programs.]

18.2.1 Complete Command File Example for SAGEM

```
!_____________________________________________________________________
!
! The following GEMPACK Command file will run SAGEM to carry out
!  a simulation with Stylized Johansen, specifying the
!  standard closure and shocking labour supply by 10 per cent.
! Only macro variables and those with one argument ranging over
!  the set SECT or FAC are retained on the Solution file.
!
use equat file  sj ;
solut file = sjlb ;    ! usually omitted – see section 2.5.1
!
! Closure
!
exogenous p_xfac ;
rest endogenous ;
save env file sjxfac ;
!
! Shocks
!
shock p_xfac 1  = 10 ;
!
! Cumulatively-retained endogenous
!
cum endog %macro (SECT) (FAC) ;
!
! Verbal description
!
verbal description =
This is a simple sim with SJ; Standard closure and data.
Labor supply increased by 10 per cent. ;
!
! Options (just examples)
!
log file = yes ; ! Log file name taken from Command file
                 !    name (see section 2.5
           ! output goes to log file and also to terminal
CPU = yes ;           ! report CPU times
!_____________________________________________________________________
```

19. References

Atkinson, Kendall E. (1989), *An Introduction to Numerical Analysis*, second edition, Wiley, New York.

Bach, C.F. and K.R. Pearson (1996), 'Implementing Quotas in GTAP using GEMPACK - or How to Linearize an Inequality', *GTAP Technical Paper* No.4, November 1996, pp.37+4.

Dirkse, S.P. and M.C. Ferris (1995), 'The PATH Solver: A Non-Monotone Stabilization Scheme for Mixed Complementarity Problems', *Optimization Methods and Software*, vol. 5, pp. 123-156.

Dixon P.B., B.R. Parmenter, J. Sutton and D.P. Vincent (1982), *ORANI: A Multisectoral Model of the Australian Economy*, North-Holland, Amsterdam.

Dixon, P.B., B.R. Parmenter, A.A. Powell and P.J. Wilcoxen [**DPPW**] (1992), *Notes and Problems in Applied General Equilibrium Economics*, North-Holland, Amsterdam.

Dixon, P.B. and M.T. Rimmer (2001), 'Dynamic, General Equilibrium Modelling for Forecasting and Policy: a Practical Guide and Documentation of MONASH', (Draft, May 2001), Centre of Policy Studies and Impact Project, Monash University. Selected chapters are available from *http://www.monash.edu.au/policy/monmod.htm*

Dixon, P.B. and M.T. Rimmer (2002), *Dynamic General Equilibrium Modelling for Forecasting and Policy: a Practical Guide and Documentation of MONASH*, North-Holland, Amsterdam.

Duff I.S. (1977), 'MA28 - A Set of FORTRAN Subroutines for Sparse Unsymmetric Linear Equations', Harwell Report R.8730 (HMSO, London), pp.104.

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.

Elbehri, Aziz and K.R. Pearson (2000), 'Implementing Bilateral Tariff Rate Quotas in GTAP using GEMPACK', *GTAP Technical Paper* No. 18, December 2000, pp. 37+4.

Ferris, M.C. and C. Kanzow (2002), 'Complementarity and Related Problems: A Survey', in P.M. Pardalos and M.G.C. Resende (ed), *Handbook of Applied Optimization*, pp. 514-530, Oxford University Press, New York, New York.

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., Mark Horridge, K.R. Pearson and Glyn Wittwer (2002), 'A Practical Method for Explicitly Modeling Quotas and Other Complementarities', *Centre of Policy Studies and the Impact Preliminary Working Paper* No. IP-78, Melbourne (April), pp.19.

Harrison, W.J. and K.R. Pearson (1996), 'Computing Solutions for Large General Equilibrium Models Using GEMPACK', *Computational Economics*, vol. 9, pp.83-127. [A preliminary version was *Impact Preliminary Working Paper* No. IP-64, Monash University, Clayton (June 1994), pp.55.]

Harrison, Jill and Ken Pearson (2002), 'Adding Accounting-Related Behaviour to a Model Implemented Using GEMPACK', *Impact Computing Document* No. C12-01, June 2002.

Hertel, Thomas W. (ed) (1997), *Global Trade Analysis: Modeling and Applications*, Cambridge University Press.

Horridge, J.M. (1993), 'Inequality Constraints', unpublished manuscript presented to the GEMPACK Users Day in June 1993.

Horridge, J.M., B.R. Parmenter and K.R. Pearson (1993), 'ORANI-F: A General Equilibrium Model of the Australian Economy', *Economic and Financial Computing*, vol.3, pp.71-140.

Horridge, Mark and Ken Pearson (2002), 'Hands-on Computing with RunGTAP and GEMPACK to Introduce GTAP and GEMPACK', July 2002. [This is the "Hands-on document" used in the 2002 GTAP Short Course held at the University of Sheffield.]

Malakellis, Michael (1994), 'ORANI-INT: An Intertemporal GE Model of the Australian Economy', PhD Thesis, Centre of Policy Studies and Impact Project, Monash University.

Malakellis, Michael (2000), *Integrated Macro-Micro-Modelling under Rational Expectations: With an application to Tariff Reform in Australia*, Physica-Verlag.

Pearson, K.R. (1988), 'Automating the Computation of Solutions of Large Economic Models', *Economic Modelling*, vol.5, pp.385-395. [A preliminary version was *Impact Preliminary Working Paper* No. IP-27, Melbourne (March 1986), pp.28.]

Pearson K.R. (1991), 'Solving Nonlinear Economic Models Accurately via a Linear Representation', *Impact Preliminary Working Paper* No. IP-55, Melbourne (July), pp.39.

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

Rutherford, T. (1993), 'MILES: A Mixed Inequality and Nonlinear Equation Solver', Working Paper, Department of Economics, University of Colorado.

Stewart, G.W. (1973), *Introduction to Matrix Computations*, Academic Press, London.

Watson, L.T. (1986) 'Numerical Linear Algebra Aspects of Globally Convergent Homotopy Methods' *SIAM Review* 28,(1985) pp529-545.

Wendner, Ronald (1999), 'A Calibration Procedure of Dynamic CGE Models for Non-Steady State Situations Using GEMPACK', *Computational Economics* vol. 13, pp.265-287.

20. GEMPACK DOCUMENTS[210]

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

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

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

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

Harrison, W.J. and K.R. Pearson (2002), *Installing and Using the Source-Code Version of GEMPACK on Windows PCs with Lahey Fortran*, GEMPACK Document No. 6 **[GPD-6],** Monash University, Clayton, Eleventh edition, October 2002, pp.39+7.

Harrison, W.J. and K.R. Pearson (2002), *Installing and Using the Executable-Image Version of GEMPACK on Windows PCs*, GEMPACK Document No. 7 **[GPD-7],** Monash University, Clayton, Eighth edition, October 2002, pp.33+6.

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

[210] The numbering of GEMPACK Documents has been re-started with Release 5 of GEMPACK, when the abbreviation "GPD" was first used. Previous editions of these documents did not have the same numbers as the current editions. Pre-Release-5 documents are numbered "GED-xx".

21. INDEX

<

<cmf> in Command files, 2-6, 2-8, 18-227

A

Abbreviation
  Command file syntax, 2-9
Acceptable Range
  Coefficient, 6-74
ACCUM, 11-136
  Documentation. *See* chapter 10 of GPD-4
Accuracy, 7-94, 12-137
  Automatic, 7-98
  Different methods, 12-140
  Iterative refinement, 12-138
  Required figures, 7-94
  Reuse of pivots, 12-140
  Subintervals, 7-96
  User-specified, 7-98
  Warnings, 12-144
Accuracy of arithmetic, 6-83
Accuracy Summary. *See* also Extrapolation Accuracy Summary
  Overall, 7-99. *See* Overall Accuracy Summary
Accurate run
  Complementarity, 16-169
Accurate run (complementarity)
  Omitting, 16-184, 16-186
Actions
  GEMSIM and TABLO-generated program, 6-61
  Options, 14-154
Actual data file
  Connection with logical file, 4-17
ADD_HOMOTOPY qualifier, 7-106, 7-108
Additional shocks, 5-48
Allowed file names. *See* chapter 5 of GPD-1
AnalyseGE. *See* chapter 2 of GPD-4
AnalyseGE example
  CVL file, 6-85
  Data-manipulation TAB file, 6-85
  Simulation. *See* section 6.1 of GPD-8
Approximate run
  Complementarity, 16-169
Arithmetic
  How accurate?, 6-83
Arithmetic errors, 15-166
Arithmetic overflow
  Meaning, 15-166
Arithmetic problems
  Analysing, 8-121
Ashock statements, 5-48
Asian characters
  Command, Stored-input files, 2-10
Assertions, 6-62, 6-63, 6-72
Automatic accuracy, 7-98, 18-229. *See also* User-specified accuracy
  Checking progress, 7-101
  Do not be too ambitious, 7-89
  Introductory example, 7-89
  Singular matrix, 7-99
  Two motivations, 7-98
  When to use?, 7-88
Automatic Accuracy Log file, 7-101
Auxiliary files
  Check, 3-14
  GEMSIM, 3-11
  Names, 3-13
  TABLO-generated program, 3-12
Auxiliary Statement file, 3-12, 3-15, 18-232
Auxiliary Table file, 3-12, 3-15, 18-232
AVC files, 8-119
Average coefficients, 8-119

B

Bad headers
  Ignore in ViewHAR, 8-121
Balance of data base
  Checking, 6-84
Base Coefficient Values file, 12-142. *See* BCV file
BCV file, 9-123, 9-125
  Introduction, 9-123
  Use not supported in next release ?, 9-125
  Using, 9-125

C

C matrix, 10-132
Centre of Policy Studies. *See* chapter 1 of GPD-1
CHANGE
  Command file statement, 5-52
Change differentiation, 7-112
Characters
  Control or foreign, 2-10
Check
  ASSERTION, 6-72
Checking balance, 6-84
Checking closure, 5-58
Checking shocks, 5-58
Check-on-read
  defaults, 4-24
Check-on-read options
  CRN,CRW,CRF, 4-24
Check-on-read statements, 4-23
Choice of variables
  Command file, 5-38
  Interactive, 5-42
  To shock or print, 5-38
Choosing variables interactively. *See* chapter 17 of GPD-4
Closure, 5-29
  Command file syntax, 18-234
  Different shocks, 10-132
  Initial check, 5-35, 5-58
  Meaning, 5-29
  Modifying commands, 5-33
  Saving, 5-32
  Sets, 6-80
  Specifying, 5-31
  Valid, 5-29

When not valid, 5-35
When processed, 6-67
CMBHAR
Documentation. *See* chapter 4 of GPD-4
CMF file. *See* Command file
CMFSTART file, 4-25, 5-44, 13-150
CMPHAR
Documentation. *See* chapter 4 of GPD-4
Coefficient initialisation, 6-82
Coefficient labelling information, 4-21
Coefficients
Acceptable range, 6-74
Zero, 12-143
Column order, 6-65, 14-155
Columns
Solution, 10-127
Command file, 4-21, 14-154, 18-227
<cmf>, 2-6
<cmf> for file names, 2-6, 2-8
Closure, 5-31
Correcting errors, 2-10
Display file name, 2-7
Eliminating syntax errors, 2-10
Example, 18-243
Extra statements, 6-79, 6-81
Length of lines, 2-8
Levels variable names, 5-59
Log file name, 2-7
Minimum subinterval statements, 7-100
Name, 2-5
On command line. *See* chapter 5 of GPD-1
Order of statements, 2-9, 5-33
Purpose of "file" statement, 4-17
Range test, 6-78
SAGEM, 5-40
SAGEM example, 18-249
Solution file name, 2-7
Statements, 17-224
TABLO-like statements, 6-79, 6-81
Variable components, 2-9
Command file stem, 2-7
Command file syntax
Abbreviation, 2-9
Automatic accuracy, 18-229
Case independent, 2-8
Choice of variables, 5-38
Closure, 18-234
Files, 18-231
SAGEM, 18-244
Shocks, 5-43, 18-236
TG Programs and GEMSIM, 18-228
Verbal description, 18-235
Command prompt. *See* section 2.3 of GPD-1
Comments, 6-79
In Command files, 2-9
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
Complementarity
Accurate run, 16-169
Approximate run, 16-169
Closure and shocks, 16-182
Closure changes for accurate run, 16-186
Command file statements, 16-184, 18-238
Dummy variable, 16-188
Examples, 16-192
Extra variables, 16-187
Newton correction, 16-169
Newton-correction variable, 16-188
Omit accurate run, 16-184, 16-186
Simulation, 16-167
State changes, 16-201
States, 16-168
Component list
Shocks, 18-236
Component numbers, 5-37
CONLF. *See* chapter 15 of GPD-4
Control characters
Command, Stored-input files, 2-10
Convergence, 12-140
ConvHAR. *See* chapter 15 of GPD-4
Correcting errors
In Command file, 2-10
CPU time, 2-8, 14-156
SAGEM, 14-156
Cumulative results, 10-129
Cumulatively-retained endogenous
Choice of, 5-38
Meaning, 8-115, 10-129
CVL file
Example, 6-85
From data manipulation TAB files for AnalyseGE, 8-120
When a simulation crashes, 8-121

D

D1C
Display file - 1-dimensional arrays, 4-22
Data
Extrapolation Accuracy Summary, 7-99
Is it balanced?, 6-84
Transferring to/from GAMS. *See* chapter 16 of GPD-4
Data files
Command file syntax, 18-230
Intermediate, 4-21
Data manipulation
Actions, 6-62
Data manipulation TAB file
CVL files for AnalyseGE, 8-120
Debug options, 18-242
Decomposing results, 11-135
Applications, 11-135
Default filenames, 2-7
Default options
GEMSIM, TG programs, 14-153
Defaults
Command file, 2-9
Demonstration Version. *See* chapter 1 of GPD-1
Description
Subtotal, 10-131
DEVIA
Documentation. *See* chapter 10 of GPD-4
Display decimals, 4-22
Display file name
In Command file, 2-7, 4-21
Display files, 4-21
Options, 4-22
Display length, 4-22
Display width, 4-22
Displays, 6-62, 6-63
All steps, 14-155

Command file, 6-79
Divide by zero errors, 15-159
Division by zero
Analysing, 8-121
Dixon, Peter, 7-107
DOI
Display file - "identical" messages, 4-22
DPN
Display file - new pages?, 4-22
Dummy variable for a Complementarity, 16-188
Duplicate headers, 18-242
Dynamic models, 5-48

E

Echo activity, 6-64
Elasticity, 10-127
Element names
Reading, 4-25
Endogenous, 5-29
Individual, cumulative, 10-128
End-point
Gragg's method, 12-140
Environment
Meaning, 5-29
Environment file
Saving, 5-32
Using, 5-32
Equations
Scaling, 12-148
Solving, 12-137, 12-138
Equations file, 9-123, 12-142
Command file syntax, 18-231
Creation, 6-61
Introduction, 9-123
Using with BCV file, 9-125
Equations Matrix, 9-124, 10-132
Equations not satisfied very accurately, 14-156
Very serious, 12-145
Error
Extra statement, 6-80
Error handling. *See* chapter 5 of GPD-1
Error trapping
Invalid argument in function, 15-165
Invalid power, 15-165
Errors, 15-159
Arithmetic, 15-166
Overflow, 15-166
Errors in Command file
Eliminating, 2-10
Euler's method, 6-65, 12-139
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
Exogenous, 5-29
Individual, cumulative, 10-128
Export quotas, 16-205
Extra data file
Command file syntax, 18-230
Extra statements, 6-79, 6-80
Extra TABLO-like statements, 6-79
Qualifiers, 6-81
Extrapolation, 6-65
Subintervals, 7-96
Extrapolation Accuracy file, 7-98, 12-146
Codes (CX etc), 7-94
Details, 7-93
Variables, 7-95
Extrapolation Accuracy Summary, 7-94, 7-98, 7-99, 12-140, 12-146
Codes (CX etc), 7-94
Data, 7-92
Number of accurate figures, 7-91
Variables, 7-91, 7-93

F

FAQs. *See* chapter 1 of GPD-1
File
LU decomposition, 12-142
Parameter, 4-20
File names
Allowed. *See* chapter 5 of GPD-1
Spaces, 2-5
File statements in Command files
Purpose, 4-17
File suffixes. *See* chapter 5 of GPD-1
File types. *See* chapter 5 of GPD-1
Filename
Default, 2-7
FINAL_LEVEL
Command file statement, 5-50
Flexible style
Element names, 4-25
Forecasting models, 5-48
Foreign characters
Command, Stored-input files, 2-10
FORMULA(INITIAL)
Save updated values, 7-110, 14-154
Frequently Asked Questions. *See* chapter 1 of GPD-1
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. *See* chapter 1 of GPD-1
Frequently Asked Questions. *See* chapter 1 of GPD-1
Hands-on introduction. *See* GPD-8
Source-code Version. *See* chapter 1 of GPD-1
GEMPACK Examples
Associated files. *See* chapter 1 of GPD-8
GEMPACK licence file. *See* chapter 1 of GPD-1
GEMPACK programs
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. *See* chapter 1 of GPD-1
GEMPIE
Choice of variables, 5-38
Documentation. *See* chapter 7 of GPD-4

Print file, 10-129
GEMSIM
Command file syntax, 18-227
Equations file, 12-143
Options menu, 14-153
GEMSIM and TG programs
Command file example, 18-243
Options. *See* Options
GEMSIM Statement file, 3-11
GEMSIM Table file, 3-11
Gragg's method, 6-65
Graphical explanation, 12-139
Number of passes, 12-140
Unsuitable in some cases, 12-146, 12-148
GTAPVIEW
Checking GTAP data, 6-83
Checking updated data, 6-83
GTAPVIEW TAB file, 6-83
GTEM model, 5-48

H

Hands-on examples. *See* GPD-8
HAR file. *See* Header Array file
Harwell Laboratories, 12-137, 12-138
Harwell parameter, 12-137
Harwell subroutine, 12-143
Header Array files
Introduction. *See* section 3.1 of GPD-4
Header Array shock files, 5-45
Homogeneity, 15-164
Nominal. *See* chapter 13 of GPD-4
Real. *See* chapter 13 of GPD-4
Homogeneity simulations. *See* chapter 13 of GPD-4
Homotopy
Example, 7-113
Homotopy method, 7-105
Homotopy terms, 7-105
Homotopy variable, 7-105
Horridge, Mark, 7-105, 16-167, 16-218
How many subintervals?
Checking the log file, 7-89

I

Ill-conditioned matrix, 12-145
Impact Project. *See* chapter 1 of GPD-1
Import quota, 16-208
Example, 16-174, 16-180
Individual column results, 10-127
Choice of, 5-38, 5-40
GEMPIE, 8-117, 10-129
Not available in multi-step simulation, 10-132
Not available via GEMSIM, 10-132
Not available via TABLO-generated program, 10-132
Not available via ViewSOL, 8-117, 10-129
Printing, 10-132
Individually-retained, 10-127
Choice of, 5-38, 5-40
Inequality
Complementarity, 16-167
INF file. *See* Information file
Input files, 4-18
In-quota tariff, 16-192
Installing Executable-image GEMPACK
On Windows PCs. *See* GPD-7
Installing Source-code GEMPACK
On Windows PCs. *See* GPD-6
Intermediate extra data file, 4-26
Intermediate file
Command file syntax, 18-230
Internediate data files, 4-26
Intertemporal models, 7-106. *See* chapter 7 of GPD-2
Invalid power, 15-165
Iterative refinement, 12-138

J

Johansen simulation, 10-127
Johansen's method, 6-65

K

Keyword
Extra statement, 6-79

L

Lahey Header Array files. *See* chapter 15 of GPD-4
Left Hand Side Matrix. *See* LHS Matrix
Levels equations
Newton's method, 7-102
Levels results, 8-117
GEMPIE, 8-117
ViewSOL, 8-117
Levels variable
Acceptable range, 6-74
Command file, 5-59
Zero, 12-140
LHS Matrix, 10-127, 12-137
Rank, 15-161
Rank deficiency, 15-162
Structural rank, 15-162
Structurally singular, 15-161
Limited Executable-image Version. *See* chapter 1 of GPD-1
Line length
In Command files, 2-8
In GEMPACK text data files. *See* section 6.2 of GPD-4
In TABLO Input files. *See* section 4.1 of GPD-2
Linearized equations
Not satisfied by accurate results, 6-70
Lists
Choice of variables, 5-38
Log file name
In Command files, 2-7
LOG files, 2-8
On command line. *See* chapter 5 of GPD-1
Log only, 2-8
Logical file, 4-17
Connection with actual file, 4-17
Long names
On updated data files, 4-21
Lower bound
Complementarity, 16-172
LTG
For compiling and linking. *See* section 1.2 of GPD-2
LU decomposition, 6-67, 12-137
LU file, 5-33, 9-125, 9-126, 12-142
Meaning, 9-126

M

MA28, 12-137
MA48, 12-138
Macro variables, 5-39
Matrix
  Equations, 10-132
  Singular, 12-145
MAX
  Complementarity, 16-210
McDougall, Robert, 3-12
Memory
  Insufficient, 13-149
Memory management. *See* chapter 5 of GPD-1
Memory required
  By GEMSIM, 13-151
  By TABLO-generated program, 13-151
  Relating to MMNZ,MMNZ1, 13-151
Midpoint method
  Can be used with -100 percent shock, 12-140
  Graphical explanation, 12-139
  Number of passes, 12-140
  Unsuitable in some cases, 12-146, 12-148
MIN
  Complementarity, 16-210
MIN file. *See* Model Information file
Miniature ORANI
  Example, 5-30
MKHAR
  Documentation. *See* chapter 11 of GPD-4
MKSOL
  Documentation. *See* chapter 11 of GPD-4
MMNZ and MMNZ1, 13-150
  Memory required, 13-151
Model identifier, 9-124
Model Information file, 3-14
Model name, 9-124
Model version number, 9-124
MODHAR
  Documentation. *See* chapter 3 of GPD-4
Modified midpoint method, 12-140
Modifying a closure, 5-33
MONASH model, 5-48
Multi-step simulation, 6-65
Multi-step solution, 12-137
  No individual column solutions, 10-132

N

Names of Auxiliary files, 3-13
Newton correction terms, 7-103
Newton correction variable, 16-169, 16-188
Newton variable
  $del_newton, 7-102
Newton's method, 7-102, 7-105
Nominal homogeneity. *See* chapter 13 of GPD-4
Nonlinear, 7-96, 12-140
Number of steps, 12-140
Numerically singular, 12-145, 15-160

O

On-quota tariff, 16-193
Options
  ACCUM - ACC and SUB, 11-136
  GEMSIM and TG programs, 14-154
  GEMSIM and TG programs - CPU, 14-156
  GEMSIM and TG programs – CPU, 2-8
  GEMSIM and TG programs - CR, 4-24
  GEMSIM and TG programs - CRF, 4-24
  GEMSIM and TG programs - CRN, 4-24
  GEMSIM and TG programs - CRW, 4-24
  GEMSIM and TG programs - D1C, 14-155
  GEMSIM and TG programs - DDC, 14-155
  GEMSIM and TG programs - DOI, 14-155
  GEMSIM and TG programs - DPL, 14-155
  GEMSIM and TG programs - DPN, 14-155
  GEMSIM and TG programs - DPW, 14-155
  GEMSIM and TG programs - DTO, 14-155
  GEMSIM and TG programs - DWS, 14-155
  GEMSIM and TG programs – DWS, 6-64
  GEMSIM and TG programs - EAA, 14-156
  GEMSIM and TG programs – EAA, 6-64
  GEMSIM and TG programs - IZ1, 12-143, 15-160
  GEMSIM and TG programs – IZ1, 12-144
  GEMSIM and TG programs - KZ2, 12-144, 15-160
  GEMSIM and TG programs - M28, 12-138
  GEMSIM and TG programs - NDS, 14-154
  GEMSIM and TG programs - NEL, 14-155
  GEMSIM and TG programs - NEQ, 14-154
  GEMSIM and TG programs – NIR, 12-138
  GEMSIM and TG programs - NRP, 12-140
  GEMSIM and TG programs - NSC, 14-155
  GEMSIM and TG programs - NSE, 14-155
  GEMSIM and TG programs - NSM, 14-154
  GEMSIM and TG programs - NUD, 14-154
  GEMSIM and TG programs - NWE, 12-145
  GEMSIM and TG programs - NWR, 14-154
  GEMSIM and TG programs - NWT, 7-102
  GEMSIM and TG programs - RQF, 7-94
  GEMSIM and TG programs - SSI, 7-96, 14-154
  GEMSIM and TG Programs - SUI, 7-110, 14-154
  GEMSIM and TG programs – SVX, 7-95
  GEMSIM and TG programs - TWC, 6-65, 14-155
  Help, 14-153
  SAGEM, 14-156
  SAGEM - CMF, 14-156
  SAGEM - Command file, 18-248
  SAGEM - CPU, 14-156
  SAGEM - KZC, 14-157
  SAGEM - NIR, 14-156
  SAGEM - NSM, 14-156
  SAGEM - NWE, 14-156
  SLTOHT - SSS, 11-136
  TABLO - ACD, 12-147
  TABLO - NTX, 8-117
  TABLO – NWT, 7-102
  TABLO-generated programs - NAX, 14-154
Options list
  GEMSIM and TG programs, 18-239
Options menu
  GEMSIM, 14-153
  TABLO-generated programs, 14-153
ORANIG
  Checking balance of data, 6-84
Order
  Reads, formulas, equations, 6-65
Oscillating results, 12-146, 12-148
Out of memory, 13-149
Output files, 4-18
Outputs from SAGEM, 10-127
Overall Accuracy Summary, 7-98, 7-99
  Codes, 7-98
  Number of accurate figures, 7-98

Overflow
  Meaning, 15-166
Over-quota tariff, 16-192

P

Parameter
  Code, 15-159
Parameter files, 4-20
Passes
  Compared to steps, 12-140
  Gragg's method, 12-140
  Midpoint method, 12-140
  TABLO-generated program or GEMSIM, 6-65
PERCENT_CHANGE
  Command file statement, 5-52
Percentage change
  Large negative, 12-140
PG date problem, 3-14
Piecewise-linear functions, 16-210
Pivots, 12-140
Preliminary pass
  Set sizes, 13-149
Program options. *See* Options
Programs
  Common features. *See* chapter 5 of GPD-1
Prompts
  Choice of variables, 5-38

Q

Quota
  Complementarity, 16-167
Quotas, 16-167

R

Range checks, 6-62. *See also* Range tests
Range tests, 6-62, 6-63
Rank deficiency, 15-161, 15-162
Rank of matrix, 15-161
Reading
  Element names, 4-25
Real homogeneity. *See* chapter 13 of GPD-4
Relative file names. *See* chapter 5 of GPD-1
Repeated values, 5-46
Residual correction, 12-138
Reusing pivots, 12-140
  May not be beneficial with MA48, 12-141
Rimmer, Maureen, 7-107
Rounding error, 12-137
Row order, 6-65, 14-155
Row totals, 10-129
RunDynam, 5-49, 5-58. *See* chapter 2 of GPD-4
  CMFSTART file, 4-25, 13-150
RunGEM. *See* chapter 2 of GPD-4
  CMFSTART file, 4-25, 13-150
RunGTAP
  CMFSTART files, 4-25
RunMONASH, 5-49, 5-58
  CMFSTART file, 4-25, 13-150
RWHAR
  Documentation. *See* chapter 11 of GPD-4
RWSOL
  Documentation. *See* chapter 11 of GPD-4

S

SAGEM
  Choice of variables, 5-38
  Command file example, 18-249
  Command file syntax, 18-244
  Equations file, 12-143
  Johansen simulation, 10-127
  Levels variable names, 5-59
  Options. *See* Options
  Solution file, 10-128
  Subtotal, 10-129
SAGEM Options, 14-156
Save Updated values
  SUI, 7-110, 14-154
Saving separate multi-step solutions, 7-109
Saving updated data after each multi-step solution, 7-109
Scalar variables, 5-39
Scaling
  Equations, 12-148
Scaling equations
  Not recommended generally, 12-148
SEEHAR
  Documentation. *See* chapter 4 of GPD-4
SEENV
  Documentation. *See* chapter 12 of GPD-4
Select from file, 5-45
Set
  Command file, 6-80
Set and element labelling
  Documentation. *See* chapter 5 of GPD-4. *See* chapter 5 of GPD-4
Set element name
  Reading, 6-80
Set elements
  Reading, 4-25
Set/element information
  checking when reading data, 4-22
Setting MMNZ and MMNZ1, 13-150
Several subintervals
  Advantages, 7-96
Shock files, 5-44, 5-53
  Assignment of numbers to shocks, 5-53
  Checks on dimensions of arrays, 5-53
  Header Array, 5-45
  Select from, 5-45
  Text, 5-45, 5-46
  Text or Header Array, 5-44
Shock of -100 percent
  Midpoint method can be used, 12-140
Shock statement
  Components in increasing order usually, 5-47
Shocks
  CHANGE statement, 5-52
  Command file syntax, 18-236
  Component list, 18-236
  Final_level statement, 5-50
  For one step, 6-68
  Meaning, 5-29
  Multi-dimensional variable, 5-53
  No shock statements, 5-59
  Only some components exogenous, 5-43, 5-59
  Percent_change statement, 5-52
  Reading from a file, 5-44
  Same closure, 10-132
  Select from file, 5-45, 18-236
  Specifying, 5-43

Subintervals, 7-96
When processed, 6-67
Short file names. *See* chapter 5 of GPD-1
Simulation
Choice of variables, 5-38
SAGEM, 10-127
Singular, 15-160
Singular matrix, 5-35, 15-159
Analysing, 8-121
Automatic accuracy, 7-99
Singular model, 15-161
SLC file, 8-118
SLTOHT
Documentation. *See* chapters 8 and 9 of GPD-4
Solution
Extrapolation Accuracy Summary, 7-99
Matrix, 10-127
Simulation, 10-127
Subtotal, 10-129
Solution accuracy, 12-144
Solution Coefficients (SLC) file, 8-118
Solution file
Saving multi-step, 7-109
Taking closure from, 5-33
Solution file =
Recommendation not to include this, 8-115
Solution file name
In Command files, 2-7
Solution files
Contents, 10-127
SAGEM, 10-127
Solution method
Command file syntax, 7-90, 18-228
Verbal description, 8-115
Which to use?, 7-88
Solution time, 12-140, 12-141
Existing files, 12-142
Subintervals, 7-96
Total, 12-141
Solutions
Saving after each multi-step calculation, 7-109
Solving
Levels equations, 7-102
Source-code GEMPACK
Installing on Windows PCs. *See* GPD-6
Source-code Version. *See* chapter 1 of GPD-1
Spaces
File name, 2-5
Sparse matrix, 12-137, 12-140, 12-143
MA48, 12-138
Speed of solution, 12-140, 12-141
Existing files, 12-142
Iterative refinement, 12-138
Subintervals, 7-96
Total time, 12-141
Zero coefficients, 12-144
SSA. *See* chapter 2 of GPD-4
SSL statements in Command files, 7-109
Standard file suffixes. *See* chapter 5 of GPD-1
Start with MMNZ|MMNZ1
Command File, 13-150
State changes
Checking, 16-201
Reporting, 16-183
States
Complementarity, 16-172
Steps
Command file syntax, 7-90, 18-228
How many to use?, 7-88
Number, 12-140
TABLO-generated program or GEMSIM, 6-65
Verbal description, 8-115
Steps, passes, 12-140
STI file. *See* Stored-input file
Stored-input files. *See* chapter 5 of GPD-1
On command line. *See* chapter 5 of GPD-1
Structurally singular, 15-160
Structurally singular model, 15-161
Strucutural rank, 15-162
Subinterval redone, 7-89
Subintervals, 7-96, 7-98
Advantages, 7-96
Command file syntax, 7-90, 18-228
How many were used?, 7-89
Introductory example, 7-89
When to use several?, 7-88
Submatrix, 6-64, 6-65, 9-124, 15-164
Subset
Choice of variables, 5-38
Subtotal, 10-130
Command file syntax, 5-38, 18-238, 18-247
Description, 10-129, 10-131
Meaning, 10-129, 10-130
Printing, 10-132
Setting up via GEMPIE, 10-131
Setting up via SAGEM, 10-130
Subtotals results
Applications, 11-135
GEMPIE, 8-117, 10-132, 11-135
Meaning, 11-133
RunGEM, 11-135
RunGTAP, 11-135
SLTOHT, 11-135
ViewSOL, 8-117, 10-132, 11-135
Suffixes
For updated data, 4-20
SUMEQ
Documentation. *See* section 13.2 of GPD-4
SUMHAR
Documentation. *See* chapter 4 of GPD-4
SUP statements in Command files, 7-109
Swapping variables, 5-33
Syntax errors in Command file
Eliminating, 2-10
Systematic Sensitivity Analysis. *See* chapter 2 of GPD-4

T

TAB character
Command, Stored-input files, 2-10
TAB file. *See* TABLO Input file
Interesting use of Sets and Coefficients, 6-83
TABLO Code
Low memory, 6-81
TABLO style
Element names, 4-25
TABLO-generated program or GEMSIM
Choice of variables, 5-38
TABLO-generated programs
Actions, 6-61
Auxiliary files, 3-12
Command file example, 18-243
Command file syntax, 18-227

Compile and link on Windows PCs. *See* section 6.3 of GPD-6
Compiling and linking. *See* section 1.2 of GPD-2
Equations file, 12-143
Executable image. *See* section 1.2 of GPD-2
Options menu, 14-153
TABLO-like statements, 6-79, 6-80
Qualifiers, 6-81
TABLO-like statments. *See* Command file. *See* Command file
TABmate. *See* chapter 2 of GPD-4
Target shocks, 5-48
Tariff-rate quota
In-quota tariff, 16-192
Introduction, 16-192
Over-quota tariff, 16-192
Tariff-rate quotas, 16-167, 16-192
Terminal data
Automatic accuracy, 7-100
Terminal data files, 4-21
Terminal writes, 6-65, 14-155
Test Coefficient
Range, 6-75
Text data files
Documentation. *See* chapter 6 of GPD-4
Text shock files, 5-45
TEXTBI, 8-117
Documentation. *See* chapter 14 of GPD-4
TG program. *See* TABLO-generated program
TGMEM
Memory required, 13-151
Time
CPU, 2-8, 14-156
Transfer statements, 6-62, 6-79
Transferring data to/from GAMS. *See* chapter 16 of GPD-4
Tshock statements, 5-48

U

U parameter, 12-137
UDC files, 8-119
Unix/Command prompt. *See* section 2.3 of GPD-1
Unlimited Executable-image Version. *See* chapter 1 of GPD-1
UPDATE
Actions, 6-61
Updated coefficients, 8-119
Updated data
Saving after each multi-step calculation, 7-109
Updated data file, 4-21
Long names, 4-21
Parameter, 4-20
Upper bound
Complementarity, 16-172
User-specified accuracy, 7-98. *See also* Automatic accuracy
Checking progress, 7-101
When to use?, 7-88
Using Equations, BCV Files
Not recommended, 9-125

V

Variables
Components, 2-9
Shocked, 5-38
Variation in results between passes, 12-146
Verbal description, 8-115
Command file, 18-235
Versions of GEMPACK. *See* chapter 1 of GPD-1
ViewHAR. *See* chapter 2 of GPD-4
Bad headers, 8-121
Ignore bad headers, 8-121
ViewSOL. *See* chapter 2 of GPD-4
Does not show individual column results, 10-130, 10-132
Shows subtotals results, 10-130, 10-132

W

Warning
Equations not satisfied very accurately, 12-145
Variation in results between passes, 12-146
Wendner, Ronald, 7-104
Windows programs
RunDynam and RunMONASH, 5-49
WinGEM. *See* chapter 2 of GPD-4
Work files
Cleaning up, 12-148
May be very large, 12-148
Writes, 6-62, 6-63
All steps, 14-155
Command file, 6-79

X

XAC-retained variables, 7-95, 7-99
XDISPLAY, 6-79
XFILE, 6-79
XSET, 6-79
XSET statements
For shocks, 5-44
In closure, 5-32
XSUBSET, 6-79
XSUBSET statements
For shocks, 5-44
In closure, 5-32
XTRANSFER, 6-79
Xtransfer statements, 6-79
XWRITE, 6-79

Z

Zahariadis, Yiannis, 5-50
Zero coefficients, 12-143, 14-157