GEMPACK USER DOCUMENTATION

Release 7.0

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

First edition
October 2000

© Copyright 1985-2000. The Impact Project and KPSOFT.

First edition
October 2000

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.

We assume that you are familiar with GEMPACK document GPD-1 *An Introduction to GEMPACK.*

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)

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 Naming the Command file 2-5
2.4.1 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 2-7

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 3-11

3.3 How are the Names of the Auxiliary Files Determined? 3-12

3.4 Check that Auxiliary Files are Correct 3-13
3.4.1 Compatibility of Various Files 3-13

3.5 Carrying out Simulations on Other Machines 3-13
3.5.1 GEMPACK Licence May Be Required 3-14

4. DATA FILES, UPDATED DATA FILES AND DISPLAY FILES 4-15

4.1 Data Files and Logical Files 4-15
4.1.1 Input and Output Data files 4-16

4.2 Updated Data Files 4-17
4.2.1 Naming Updated Files 4-17
4.2.2 Updated Data Read from the Terminal 4-17
4.2.3 Intermediate Data Files 4-18

4.3 Display Files 4-18
4.3.1 Options for Display Files 4-18

4.4 Checking Set and Element Information when Reading Data 4-19
4.4.1 Check-on-Read Problems with RunGEM, RunDynam and RunMONASH 4-21
4.4.2 Checks When Reading Set Element Names from a File 4-21

4.5 Names of Intermediate Data Files 4-22
4.5.1 Default Names for Intermediate Files When Using a Command File 4-23

5. CHOOSING THE CLOSURE AND SHOCKS 5-25

5.1 Specifying the Closure 5-25
5.1.1 Miniature ORANI Model 5-25

5.2 Specifying the Closure via a Command File 5-27
5.2.1 Specifying the Closure from Scratch 5-27
5.2.2 Using a Saved Closure 5-28
5.2.3 Modifying a Saved Closure 5-29
5.2.4 Closure Specified Using Data Files 5-29
5.2.5 List of Exogenous/Endogenous Variables when Closure is Invalid 5-30

5.3 Component Numbers for a Variable 5-30
5.3.1 Examples of Component Numbers 5-30

5.4 Choosing Other Sets of Variables 5-31
5.4.1 Choosing Sets of Variables via a Command File 5-31
5.4.2 Example of Choosing Sets in SAGEM 5-33
5.4.3 Choosing Sets of Variables Interactively 5-34

5.5 Specifying the Shocks 5-35
5.5.1 Specifying the Shocks via a Command File 5-35
5.5.2 Using a Shock File 5-36
5.5.3 Reading Shocks from Header Array file 5-36
5.5.4 Shocking a Variable with No Components Exogenous 5-36
5.5.5 Additional and Target Shock (ashock or tshock) Statements for Dynamic Models 5-37

5.6 Clarification Concerning Shocks on a Command File 5-38
5.6.1 Shock Statement when only Some Components are Exogenous 5-38
5.6.2 No “Shock” Statements on a Command File 5-38

5.7 Specifying the Shocks Interactively 5-39

5.8 Shock File Syntax 5-40

5.9 Checking the Closure and Shocks 5-41

6. ACTIONS IN GEMSIM AND TABLO-GENERATED PROGRAMS 6-43

6.1 Possible Actions in GEMSIM and TABLO-generated Programs 6-43
6.1.1 Multi-Step Simulations with Economic Models 6-43
6.1.2 Creating an Equations File 6-43
6.1.3 Other Actions (Writes, Displays, Assertions, Range Tests and Transfers) 6-44
6.1.4 Data Manipulation 6-44
6.1.5 “Extra” (TABLO-like) Actions 6-45
6.1.6 Checking the Closure and Shocks 6-45
6.1.7 Controlling Whether and How the Actions are Carried Out 6-45
6.1.8 Some READs and FORMULAS May be Omitted 6-45
6.1.9 Writes and Displays at all Steps of a Multi-step Simulation 6-46
6.1.10 Echoing Activity 6-46
6.1.11 Writes to the Terminal 6-46

6.2 How These Programs Carry Out Multi-step Simulations 6-47
6.2.1 Processing the Closure and Shocks in GEMSIM or TABLO-generated Programs 6-49
6.2.2 Results of a 4-step Simulation Looked at in Detail 6-50

6.3 Assertions 6-51

6.4 Range Tests 6-52
6.4.1 How to Specify the Allowed Range for Values of a Coefficient or Levels Variable 6-53
6.4.2 Tests Carried Out 6-54
6.4.3 An Example 6-54
6.4.4 Associated Statements in Command Files 6-55

6.5 Transfer Statements 6-56

6.6 TABLO-like Statements in Command Files 6-56
6.6.1 TABLO-Like Checks of Extra Statements 6-57
6.6.2 Qualifiers 6-57
6.6.3 Other Points 6-57

6.7 Coefficients May Not Be Fully Initialised 6-58

7. MULTI-STEP SOLUTION METHODS 7-59

7.1 Command File Statements for Method and Steps 7-59

7.2 Extrapolation Accuracy Summaries for Data and for Levels Variables 7-60
7.2.1 For Data 7-60
7.2.2 For Levels Variables 7-60
7.2.3 Command File Statements Affecting Extrapolation Accuracy Summaries and Files 7-61

7.3 Splitting a Simulation into Several Subintervals 7-62
7.3.1 Advantages of Using Several Subintervals 7-64

7.4 User-specified Accuracy for Simulations 7-64
7.4.1 Specifying Which Solutions to Use 7-65
7.4.2 Incompatibilities 7-65
7.4.3 Automatic Accuracy with Models which Read Data from the Terminal 7-66
7.4.4 Adaptive Stepsize Method 7-66
7.4.5 Stopping if Subintervals Become Too Small 7-66
7.4.6 Controlling Which Values are Tested 7-66

7.5 Newton's Method for Levels Models 7-67
7.5.1 Newton's Method and ORANIF 7-67
7.5.2 Convergence 7-68
7.5.3 General advice 7-68
7.5.4 Newton's Method Is Not Reliable with Mixed Models 7-68

7.6 Options for Saving Solution and Updated Data Files 7-69
7.6.1 Saving Solutions or Updated Data After Separate Multi-step Calculations 7-69
7.6.2 Saving Updated Values from All FORMULA(INITIAL)s 7-69

8. SOLUTION FILES 8-71

8.1 Command File Statements Related to Solution files 8-71
8.1.1 SAGEM Only Command File Statements 8-72

8.2 Contents of Solution Files 8-72
8.2.1 TABLO File and TABLO STI File Stored on Solution File 8-73

8.2.2 Command File Stored on Solution file 8-73

8.3 Levels Results 8-73

8.4 Solution Coefficients (SLC) Files 8-74
8.4.1 Solution and SLC Files are Complete Record of the Simulation 8-74

9. EQUATIONS FILES AND BASE COEFFICIENT VALUES (BCV) FILES 9-75

9.1 Equations Files 9-75
9.1.1 Using an Equations file in SAGEM 9-75
9.1.2 Differences in Step 1 Depending on Whether or Not Equations File is Saved 9-75
9.1.3 Model Name, Version and Identifier 9-76

9.2 Base Coefficient Values (BCV) Files 9-77
9.2.1 Starting from Existing Equations and BCV Files Not Recommended 9-77

10. SEVERAL SIMULTANEOUS JOHANSEN SIMULATIONS VIA SAGEM 10-79

10.1 The Solution Matrix 10-79
10.1.1 Individual Column Results 10-80
10.1.2 Cumulative or Row Totals Results 10-80
10.1.3 Subtotal Solutions 10-80
10.1.4 Contents of a SAGEM Solution File 10-80
10.1.5 Printing the Results of a Simulation 10-81

10.2 Subtotals 10-81
10.2.1 First Method using SAGEM 10-82
10.2.2 Second Method using GEMPIE 10-82
10.2.3 Subtotals and Sets of Shocks in General 10-82
10.2.4 Printing Subtotals or Individual Column Results using GEMPIE 10-82

10.3 No Individual Results from Multi-Step Simulations 10-83

11. SUBTOTALS VIA GEMSIM OR TABLO-GENERATED PROGRAMS 11-85

11.1 Subtotals from GEMSIM and TABLO-generated Programs 11-85

11.2 Meaning of Subtotals Results 11-85

11.3 Subtotal Example 11-86

11.4 More Substantial Examples 11-87

11.5 Processing Subtotal Results 11-87

11.6 Subtotals Results Depend on the Path 11-87

12. SOLVING MODELS AND SIMULATION TIME 12-89

12.1 How GEMPACK Programs Solve Linear Equations 12-89
12.1.1 Sparse-Linear-Equation Solving Routines MA48 and MA28 12-90
12.1.2 Which to Use? MA48 or MA28 12-90

12.2 Gragg's Method and the Midpoint Method 12-91

12.3 Reusing Pivots 12-92

12.4 Time Taken for the Different Steps 12-93
12.4.1 Total Time for Multi-step Compared to 1-step 12-93
12.4.2 Using Existing Files to Speed Up Step 1 12-94

12.5 Ignoring / Keeping Zero Coefficients 12-95

12.6 Equation Solving Problems 12-96
12.6.1 Warnings About Equations Not Being Satisfied Very Accurately 12-96
12.6.2 Warnings About Variation Between Passes in Endogenous Results 12-97
12.6.3 Gragg and Midpoint Are Not Suitable in Some Cases 12-97
12.6.4 Linearization Used Can Affect Accuracy 12-98
12.6.5 Scaling Equations 12-99

12.7 Work Files 12-99

13. COMPILERS AND MEMORY MANAGEMENT 13-101

13.1 Fortran 90 Compilers 13-101
13.1.1 Automatic Memory Allocation with Fortran 90 13-101
13.1.2 Preliminary Pass 13-102
13.1.3 Repetitions if Memory Allocation is Too Small 13-102
13.1.4 Long File Names Available on Windows PCs with Fortran 90 13-102
13.1.5 Reporting Memory Used by TABLO-generated Programs and GEMSIM 13-103
13.1.6 Choice of Fortran 77 or Fortran 90 TABLO-generated Programs 13-103

13.2 Fortran 77 Compilers 13-104
13.2.1 No Long File Names or Spaces with F77L3 13-104
13.2.2 Increasing Parameters in Fortran 77 Main Programs 13-104

13.3 Increasing Parameters for TABLO-generated Programs and GEMSIM 13-105
13.3.1 Parameters MMNZ and MMNZ1 13-105
13.3.2 Increasing Parameter Values in GEMSIM 13-105
13.3.3 Increasing Parameter Values for a TABLO-generated Programs 13-106

14. OPTIONS FOR GEMSIM AND TABLO-GENERATED PROGRAMS 14-109

14.1 Options Affecting Simulations 14-110

14.2 Options Affecting Extrapolation Accuracy Summaries and Files 14-110

14.3 Saving Updated Values from All FORMULA(INITIAL)s 14-110

14.4 Options Affecting the Actions Carried Out 14-110

14.5 Options Affecting How Writes and Displays are Carried Out 14-111

14.6 Options Affecting CPU and Activity Reporting 14-111

14.7 Special Options for SAGEM 14-112
14.7.1 SAGEM Options 14-112

15. RUN-TIME ERRORS 15-113

15.1 Simulation Fails Because of a Singular LHS Matrix 15-114
15.1.1 Trying to Fix a Singular Matrix Problem 15-115

15.2 Finding Where an Arithmetic Error Occurs and Rectifying It 15-116

15.3 Division by Zero Errors 15-117

15.4 Error Trapping for Invalid Powers, Logs etc 15-117

15.5 Overflow and Other Arithmetic Errors 15-118
15.5.1 Other Sorts of Arithmetic Errors 15-118

16. SYSTEMATIC SENSITIVITY ANALYSIS AND QUOTAS 16-119

16.1 Systematic Sensitivity Analysis 16-119

16.2 Explicit Volume Quotas and Other Inequalities 16-119
16.2.1 Explicit Volume Quotas 16-120
16.2.2 Tariff Rate Quotas 16-120

17. NEW COMMAND FILE STATEMENTS 17-121

17.1 New Command File Statements for Release 7.0 17-121

17.2 New Command File Statements for Release 6.0 17-121

17.3 New Command File Statements for Release 5.2 17-122

18. SUMMARY OF COMMAND FILE STATEMENTS 18-123

18.1 Command Files for GEMSIM and TABLO-generated Programs 18-123
18.1.1 Command File Statements 18-124
18.1.2 General Points 18-137
18.1.3 Complete Example 18-139

18.2 Command Files for Running SAGEM 18-140
18.2.1 Complete Example for SAGEM 18-145

19. REFERENCES 19-147

20. GEMPACK DOCUMENTS 20-148

21. INDEX 21-149

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.

Other topics covered include memory management with a Fortran 90 compiler (see chapter 13), options for GEMSIM and TABLO-generated programs (chapter 14) and run-time errors (chapter 15). Chapter 16 describes how systematic sensitivity analysis can be carried out via GEMPACK and how quotas can be handled in GEMPACK.

[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 2.11 of GPD-1.

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

- **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 3.4.3 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 2.10 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 2.10 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 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 (or if you are using the Fortran 77 compiler F77L3 which can only handle short names).

See section 4.9 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.

2.4.1 File Names Containing Spaces

On Windows PCs (provided you are not using F77L3), file names can contain spaces. This may also be possible on Unix machines – consult your GEMPACK Manager to find out. Details about this can be found in section 4.9 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    ” ;
```

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[2]
- 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 ;
```

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 F77L3 – see section 4.9 of GPD-1.]

---

[2] 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.

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).[3] 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*). [See section 4.3.3 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***.

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:

---

[3] The suffix .CMF is not required by GEMPACK (see section 4.8.2 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.

```
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[4]

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: 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 :
  `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

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

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) 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,
  Shocks only given to domestic inputs!! Foreign ones given no shock.
- 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
```

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: The components must be specified in increasing order, as in

```
shock v1 3 4 8 10 = 1.1 4.4 3.3 2.4 ;
```

Shocks can be input using the repeated value format. For example,

```
shock v1 3 4 8 10 12-16 = 3*1.3   4.4  5*-2.1 ;
```

the shocks are 1.3 1.3 1.3 4.4 -2.1 -2.1 -2.1 -2.1 -2.1. (Don't use brackets around negative shocks.)

To shock some components of a multi-dimensional variable, remember that the first index varies fastest (see section 0) and specify component numbers accordingly. For example, for a 3x5 variable V, to shock V(2,1), V(3,1), V(2,2),

```
shock v 2 3 5 = 4.5 2.0 1.0 ;
```

- <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.

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 of GPD-3). 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.]

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 2.1.1 about TABLO-generated programs and section 2.1.2 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 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.)[5]

3.2 TABLO-generated Programs and Auxiliary Files

When you run a TABLO-generated program, the **name of the program** gives all the information necessary to say which economic model (as written down in the TABLO Input file) you are using.

> For example, if you run the TABLO-generated program SJ.EXE made from the TABLO Input file SJ.TAB, then the EXE file and its two Auxiliary files SJ.AXS and SJ.AXT store all the information about the Stylized Johansen model which was written down in the TABLO Input file SJ.TAB.

TABLO-generated programs must be able to access their associated Auxiliary Statement and Table files (with suffixes '.AXS' and '.AXT' respectively). The names of these are hard-wired in the source code of the program, and these names usually suffice.

---

[5] 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.

If you are copying the TABLO-generated program to a different directory, you should also copy its associated Auxiliary files to the same directory[6].

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).[7]

You can include **directory and path information** in the <file_name>.[8] If you are working in the same directory as the EXE file and its associated Auxiliary files, this statement is **optional.**

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 TABLO-generated program SJ.EXE and its two Auxiliary files SJ.AXS, SJ.AXT are in directory C:\SJ. To run the program for example at the command prompt, you can enter the name of the program:

c:\sj\sj.exe -cmf sjlb.cmf

but so that the TABLO-generated program finds its Auxiliary files in directory C:\SJ, you need to include in your Command files the statement:

auxiliary files = c:\sj\sj ;

If you want a Command file which can be used for either GEMSIM or the corresponding TABLO-generated program, you can include an Auxiliary files statement as above, provided that the <file_name> for the GEMSIM Auxiliary files for GEMSIM is the same as the name of the TABLO-generated program.

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

3.3 How are the Names of the Auxiliary Files Determined?

Beginning 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.

The answer is simple. The name you use in the "auxiliary files = … ;" statement in your Command file are

the names of the Auxiliary files produced when you ran 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). In either case you could put the statement
  `auxiliary files = xx ;`
  in your Command file.
- 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. In that case, you would put the statement

---

[6] If you made the TABLO-generated program and its Auxiliary files in WinGEM, you may need an "Auxiliary files" statement after moving the files to different directory, even if the Auxiliary files and the Executable image of the program are in the same directory

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

[8] If you use WinGEM to run TABLO, you should be aware that WinGEM normally puts the full path name (for example, C:\SJ\SJ) of the Auxiliary files into the TABLO-generated program. This means that you will need to use an "auxiliary files = … ;" statement if you move the program to a different directory.

```
auxiliary files = xxcon ;
```
in your Command file.

3.4 Check that Auxiliary Files are Correct

TABLO-generated programs check that the Fortran file from which the Executable Image was made 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, the old Executable Image of the TABLO-generated program does not match the new Auxiliary files.[9]

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

3.4.1 Compatibility of Various Files

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 program RunGEM – see section 2.5 in GPD-4.[10]

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 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,

- 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.

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.

To transfer a TABLO-generated program, you must copy the executable image of the program (this often has suffix '.EXE') and also copy the Auxiliary Statement and Table files (these usually have suffixes '.AXS' and '.AXT' respectively) to the other machine since the TABLO-generated program will not run without them.

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 produce human readable output from your simulation.

---

[9] 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.

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

Note that, because 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). Rather, 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 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.

3.5.1 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 1.2.5 of GPD-1.

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

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. When the carry out a simulation, they normally produce updated data files (which are post-simulation versions of the corresponding original data files).

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[11]. 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 ORANIG model, there is one input data file with logical file name MDATA. This is defined in the TABLO Input file for ORANIG 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.
>
> 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.

[11] It is possible to put the actual file name in your TABLO Input file 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.

Example 2 – Stylized Johansen

In the TABLO Input file for Stylized Johansen 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.

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 (WRITE) 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. Output files (those which are written to) are declared with the **(NEW)** qualifier as in

```
File (NEW) OutFile ;
```

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 ;
```

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 is being written to (that is, is an output file). If FILE3 corresponds to an input file, then the file f3.dat must exist before the TABLO-generated program runs, whereas if FILE3 corresponds to an output file, the file f3.dat will be created by the TABLO-generated program (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 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.

4.2 Updated Data Files

If you have a FILE statement in your TABLO Input file, and if any coefficient read from that logical file is updated using UPDATE statements (or there is a Levels variable where there is an automatic UPDATE statement), 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 the name of the new updated version of the original data file which will be created on your computer disk after the simulation. It 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.

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 PC, Unix and VAX/VMS machines), this means choosing a suffix of length 3 (following the '.'); we find the suffix **.upd** ideal.[12]

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

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 file.

---

[12] 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).

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 0 below.

4.3 Display Files

If you want to see the values of a Coefficient, 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**.

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 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.

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.[13]

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.[14]

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).

---

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

[14] 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.

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:

- 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 they think is the second sector (namely *manufactures*[15]) 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[16] 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 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.

---

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

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

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.

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 check-on-read settings) 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 coefficient = no ;
check-on-read exact = no ;
```

These are what you get if you don't put any check-on-read statements in your TABLO Input file. If you find that this is causing fatal errors when no data association errors are in fact occurring (this can happen for all sorts

of reasons), you can change to just warnings (or no checking) for some or all of the three possibilities (coefficients, sets, elements). Of course, if you do not wish to do 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.[17] 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 ;".

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 TABLO-generated program read 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 and RunMONASH

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.

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 element names that must be 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 new Command file statement to choose between these two styles[18]:

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

The default is TABLO style and we would 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 additional check.

[17] 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.

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

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 step of the calculation (this applies if option UCS has been selected in the Code stage of TABLO) and/or 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.[19]

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.

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, and/or
- option UCS has been selected from the Code Options Menu in TABLO when the program was written and there is more than one pass in one or more of the multi-step simulations.

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> ;
```

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 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.

---

[19] 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.

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.[20] 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.

[20] 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 (section 5.5). All of this 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[21] exceeds the number of equations (as we have seen in section 2.11.1 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, 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

$$z = - x – y.$$

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.[22] In section 5.2` below we discuss the different ways of specifying the closure on a Command file.

Note that, if you have levels VARIABLEs in your TABLO Input file, you must use the names of the associated linear variables (these 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).

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

[21] As explained in section 2.11.1 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.)

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

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 section 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. You can then run the program 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 5.2.4.

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. (Note that these are 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.)

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. Alternatively, components can be indicated using set and/or element names[23] 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 0 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.

For example, the following statements specify the usual closure:

```
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:

[23] 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.

```
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.

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, we 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.[24]

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 an Command file when this Solution file contains the values of variables which have been backsolved for.

---

[24] An LU file stores the closure and the LU decomposition of the left-hand side matrix (see section 7.5 below). The statement

take closure from LU file <file_name> ;

is another way of specifying the closure. This should be compared with the similar statement

use LU file <file_name> ;

which means that the LU decomposition is taken from this file, as well as the closure.

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 2.10.4 of GPD-1 and in sections 18.1.3 and 18.2.1 of this document and in the model files listed in section 1 of GPD-8.

5.2.4 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.[25] 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 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.

[25] This was introduced in Release 5.2.

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 ;
```

This method of specifying part of the closure was introduced to facilitate the closure swaps needed when handling quotas and other inequalities – see section 16.2.

5.2.5 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, the program gives a list of the exogenous variables and a list of the endogenous ones (each in Command file syntax) before exiting with an error. Users may find these lists helpful in identifying variables incorrectly specified as exogenous or endogenous. We are grateful to Mark Horridge for suggesting this.

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.

Note that, if you have levels VARIABLEs in your TABLO Input file, you must use the names of the associated linear variables (these have "p_" or "c_" added at the start – see section 2.2 of GPD-2) when choosing sets of variables.

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, we 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 [26]

   p_XINTFAC("labor",IND)

   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 components of it (these 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.

[26] 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.

5.4.2 Example of Choosing Sets in SAGEM

Suppose shocks are to be applied to the following exogenous variables (we are assuming the closure is as saved on the Environment file MO.EN4 – see the section 5.2.1) :

p_PHI, component 2 of p_T, p_FWAGE and p_CR.

The three commands in the Command file 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
!  from TABLO Input file MO.TAB
use equations file  mo ;
solution file = mosagem ;
!
! Closure
use environment file mo ;
!
! 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) ;
!
verbal description = MO Standard closure.
Shocks  p_phi p_t 2 p_fWAGE p_cR ;
```

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

The command

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

means that the Solution file will contain results 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

all endogenous macro variables (that is, variables with just one component) - 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.)

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.)

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.[27] (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.

[27] 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

If you are using a 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. 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 5.7 below. (As before, always use names of linear VARIABLEs when specifying variables to be 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.

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.

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 ;
```

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 file FACTOR.SHK into the components of variable p_XINTFAC. The method of preparing shock files is discussed in section 5.8.

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 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 0 above). The words "**select from file**" are what distinguishes 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.

5.5.3 Reading Shocks from Header Array file

In Release 7.0 (or later), the values of shocks can be read from a header on a Header Array file. [Previously, they could only be read from a text file.] The Command file syntax is as for text files except that you must add "header <header>" at the end. For example,

Shock x1 = select from file xx.har header "ABCD" :

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.

If you have a file containing the Environment (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.

5.5.4 Shocking a Variable with No Components Exogenous

A statement on Command files for GEMSIM and TABLO-generated programs (but not SAGEM) is[28]

```
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.

However, in handling inequalities (such as quotas), 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). The procedures described there apply to any model implemented using

[28] This statement was introduced in Release 6.0.

GEMPACK; GTAP is taken as the example in that paper. See also section 16.2.] 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.

5.5.5 Additional and Target Shock (ashock or tshock) Statements for Dynamic Models

If you work with recursively dynamic models such as MONASH 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 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[29].

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** and **RunGTEM**) – see section 2.7 of GPD-4.[30] 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

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

at the end (or elsewhere). The software will combine the two shocks (20% and the additional 5%) at run time. 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.[31]

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.]

---

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

[30] These windows interfaces, which are designed to automate the solution of recursively dynamic models such as MONASH, are available. Consult the GEMPACK developers for more information.

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

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 statement for the same component of the same variable. Thus, for example, the two statements

ashock xtax = uniform 10 ;
ashock xtax = uniform 1 ;

would result in an error. The following two statements

ashock xtax = uniform 10 ;
tshock xtax = uniform 1 ;

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.

5.6 Clarification Concerning Shocks on a Command File

This section aims to clarify issues concerning shocks on a Command file which were ambiguous in the previous documentation[32].

5.6.1 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, the shocks for the exogenous components of the variable will be selected appropriately from amongst the N values on the file.]

5.6.2 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.

[32] 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.

5.7 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). 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 5.8 below 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.

5.8 Shock File Syntax

If you specify a file from which to read the shocks for one variable, this file may be a GEMPACK text file or a GEMPACK Header Array file (see section 5.5.3). If it is a Header Array file, you can read shocks for several variables from the same shock file since you can use different Headers for different variables. If it is a text file, you need one file per variable shocked.

If the shock file is a text file, it must begin with the "how much data" information which, on shock files, can usually take the simple form

<number> ;

where <number> is the number of numerical values of shocks on the file and the ending semicolon ';' is required. In general, <number> is the number of shocked components of the variable in question. However when you are selecting shocks from a file using the "select from file.." statement as described in section 5.5.1, <number> is the total number of components for the variable.

Then, on the subsequent line (or lines, as you please) must come <number> real numbers, being the values of the shocks.[33]

For example, to give 5 shocks, the file could contain

```
5 ;
1.2  13.0   4   -5   7.0
```

These numbers can be separated by spaces or commas.

Repeated shocks of the same size can be abbreviated using the syntax

<integer>*<value>

so that, for example, **20*1.3** means 20 values each equal to 1.3. (There should be no space before or after the '*'.)

For example to give 100 shocks with the first 20 equal to 1.3 and the remainder equal to –15.0, the file could contain

```
100 ;
20*1.3  80*-15.0
```

Note that each such file can only contain the shocks for one variable. For each variable, if you specify a file from which to read the shocks, the file is read from the top.

To shock a multi-dimensional variable, for example X(I,j) where I varies from 1 to 3, j varies from 1 to 5 enter 15 shocks (with I running fastest) that is, in the order

```
X(1,1), X(2,1), X(3,1), X(1,2), X(2,2),….. X(3,5)
```

In this example, the file might be as below.

```
 15 ;
1.1  2.1  3.1
1.2  2.2  3.2   1.3  2.3  3.3
1.4  2.4  3.4   1.5  2.5  3.5
```

Section 5.3 explains how components of a variable with two or more arguments are ordered.

Shock files must conform to the syntax of GEMPACK text data files, as documented in chapter 6 of GPD-4.

---

[33] (i) If the variable in question has <number> shocked components, the data on the shock file can actually be that for an array of size <n1> x <n2> x... where the product of the sizes n1,n2,... equals <number>.

(ii) The "how much data" information was not required before Release 5 of GEMPACK. It is included now as a further check that the shocks are as you intend.

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 the shocks 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).[34] [Of course you must comment these out when you really want the simulation to run.]

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.

[34] 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.

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-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 section 2.11.3 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.[35]

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 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.

[35] The only example amongst the models usually supplied with GEMPACK is the DMR model (see section 1.3 of GPD-1). Of course, this could have UPDATE statements added to it, as explained in Chapter 4 of DPPW.

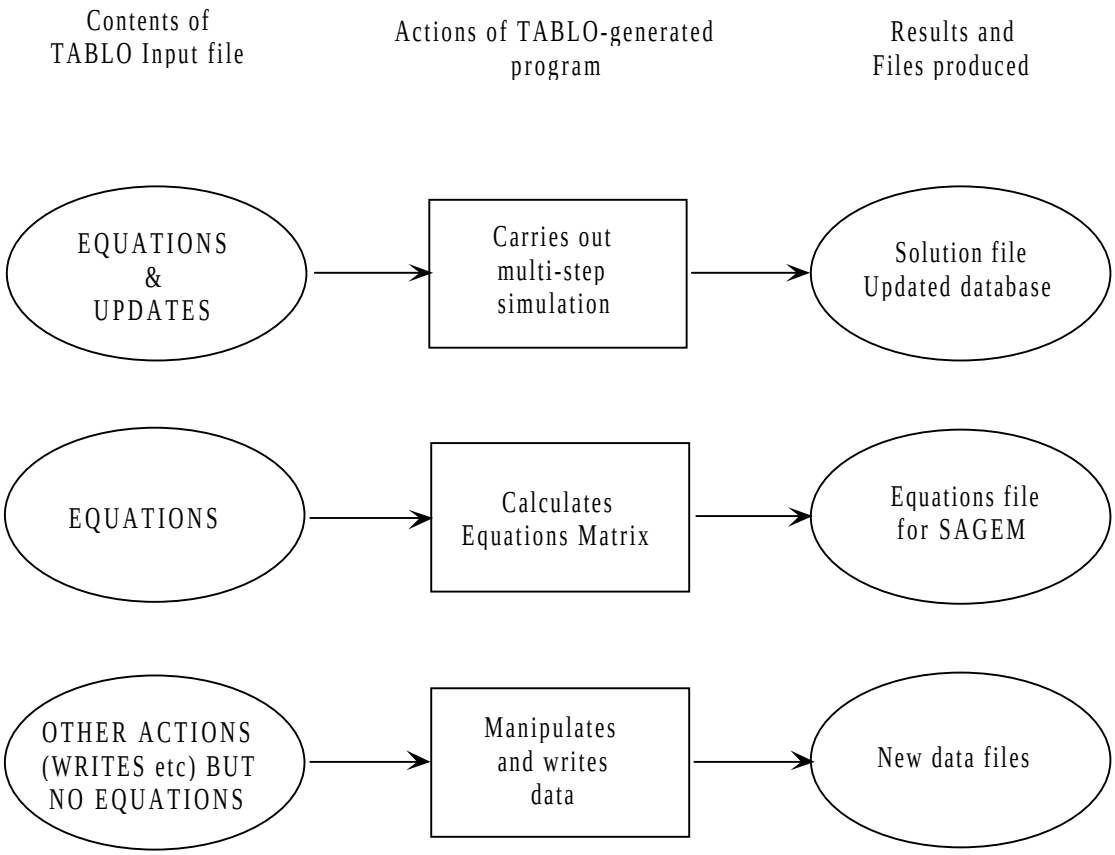


Figure 6.1: Actions of GEMSIM and TABLO-generated Programs

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 3.4.3 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.

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).

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[36]

```
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; see section 2.7.2 of GPD-1 for an example of this.

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 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 ;
```

---

[36] 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).

```
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[37]

`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[38]

`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 put the statement[39]

`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[40]

`twc = yes ;` ! Terminal Writes in Column order

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.

---

[37] Alternatively, you can select option DWS (see section 14.5).

[38] Alternatively, you can select option EAA (see section 14.5).

[39] Alternatively, you can de-select option EAA [that is, respond **-EAA**] (see section 14.5).

[40] Alternatively, you can select option TWC (see section 14.5).

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,
2. calculation of equations (one submatrix at a time),
3. the simulation part,
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 4.14.6 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.[41]

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 (I) 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, 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.]

[41] The output from GEMSIM and TABLO-generated programs distinguishes between "steps" and "passes". Thus a 6-step Gragg does 7 passes.

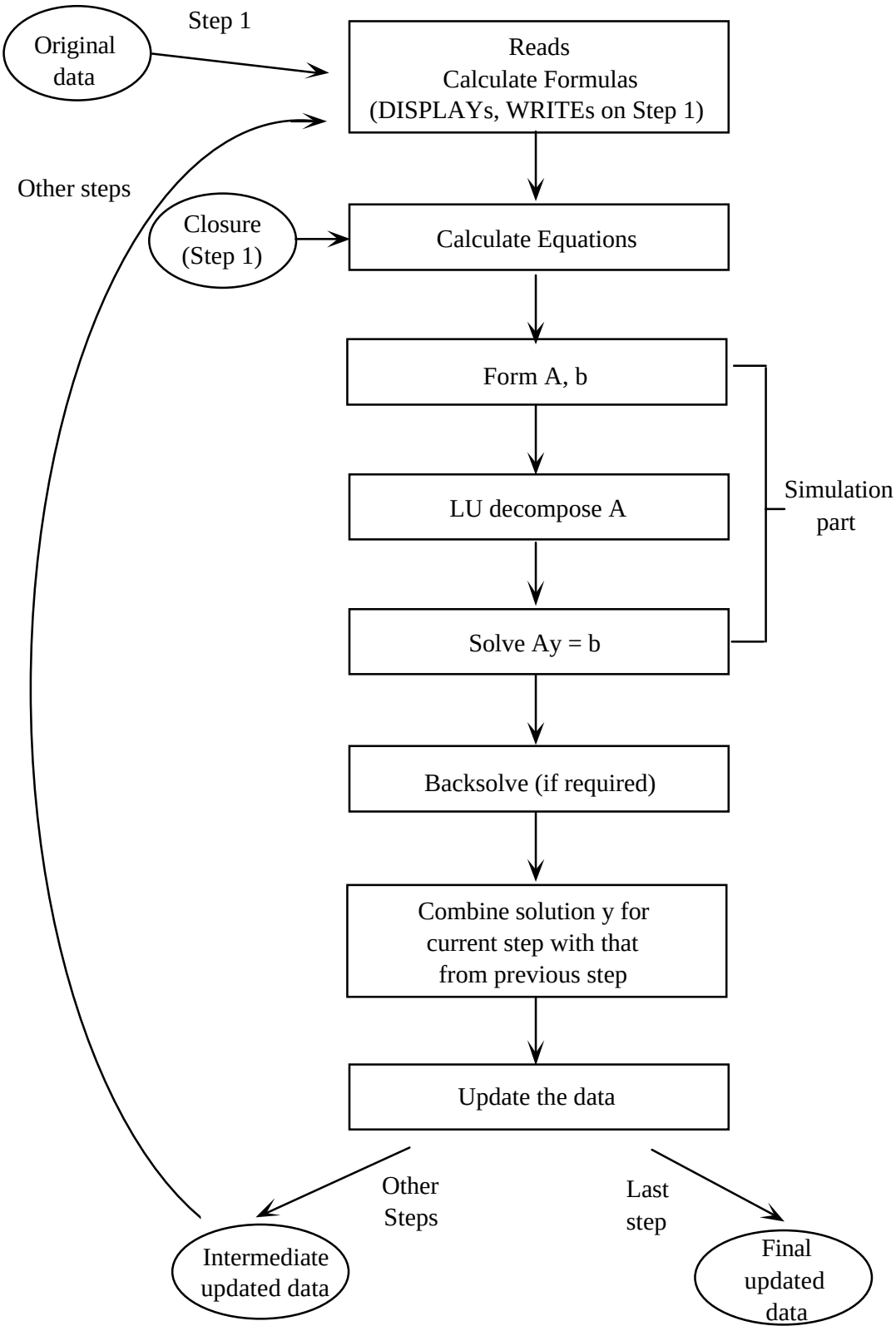


Figure 6.2: Calculation of One Multi-step Solution

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 2.11.1 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 2.11.2 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. 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 process the closure as soon as all SET and SUBSET statements (including any "extra" ones – see section 6.6) have been processed, and 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[42].

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.

[42] This is earlier than they processed the closure and shocks in Release 5.1.

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 2.2 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 well 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.[43]

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.[44]

In the second step, in response to the 2.439% shock to XF("s1"), 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%.

---

[43] 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 2.2.1a if the price is one dollar per unit).

[44] 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 2.11.3 of GPD-1) since the shock is exactly one quarter.

In the third step, in response to the 2.381% shock to XF("s1"), 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 XF("s1"), 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 XF("s1")] 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 2.11.3 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 XF("s1")] is 6.925%.

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.[45] 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.
```

[45] This was new for Release 6.0. [Release 5.2 software did not show this.]

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.[46]

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.

6.4 Range Tests

Coefficients and levels variables may naturally have a certain range of acceptable values. For example, a certain coefficient (perhaps one representing a dollar value) 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 (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 is 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.

[46] The Command file statement "NAS = yes ;" is an alternative to "assertions = no ;". This corresponds to the interactive option NAS shown when GEMSIM and TABLO-generated programs run.

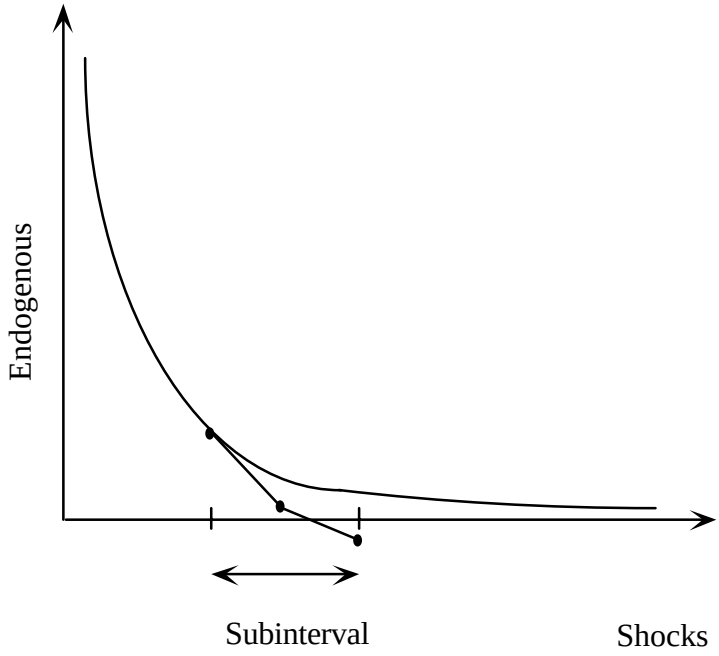


Figure 6.4 : Levels value can become negative if subinterval is too long

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.

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).

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.

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.

If TABLO Code option LMC (Low-Memory Code) is selected, the checking is 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).

(b) Test of Updated Values

These range qualifiers are used to test the UPDATEd value 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.

6.4.3 An Example

Consider the experiment called E1 in the GTAP application about the benefits of abolishing the Multi-Fibre Agreement (see chapter 10 of the GTAP book, Hertel (1997)). If this is solved using Euler 2,4,6-steps, the value of VXMD("CLG","AUS","NAM") at the end of the first 2-step calculation is negative. Yet the results give no indication that a negative value has been used in extrapolating to the final updated value of this part of VXMD.[47]

---

[47] The results of this application, as reported in chapter 10 of Hertel (1997), are not affected by negative VXMD values since they (wisely) used a Gragg 4,6,8-step calculation for experiment E1.

Even with a Gragg 4,6,8-step calculation, the value of VXMD("CLG","AUS","NAM") is negative after the third step of the Gragg 4-step calculation. This also is not obvious unless a "GE 0" qualifier is effective. [However, the results obtained using the "GE 0" qualifier are not significantly different from those reported in chapter 10 of Hertel (1997); of course, the authors of that chapter did not have access to these "GE 0" qualifiers.] The Command file for this experiment is the one usually called C2-06E1.CMF.

If this same simulation were carried out with a "GE 0" qualifier for VXMD and user-specified accuracy was used, the software would recognise this value and automatically adjust the step size to avoid having any negative values during the calculations and extrapolations. If user-specified accuracy was not used, at least the error (that VXMD goes negative) would be reported and you would be able to rerun it with more steps and/or subintervals.

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) (*both* is the default)
- *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,[48] 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 warnings 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.

---

[48] 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**[49]. 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) is used instead of the TABLO Input file syntax for comments (in which text between two exclamation marks ! is a comment). [TABLO-like comments consisting of text between two exclamation marks '!' or between strong comment markers '![[!' and '!]]!' are not allowed in extra statements on Command files.]

There are two main reasons why we now 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, (see Figure 2.8.1 of GPD-1) to examine the values of coefficient ALPHAFAC, the following statements could be added to the Command file SJLB.CMF:

   ```
   xfile (new,text) sjout ;
   xwrite alphafac to file sjout ;
   file sjout = SJLBOUT.DAT ;
   ```

2. To allow you to define SETs which may be useful in specifying a specific closure of your model. 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 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) ;
   ```

[49] TABLO-like statements in Command files were introduced in Release 5.2.

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'.

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") ;
```

Similarly you can define extra sets to specify neatly which components to shock in a particular simulation.

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. 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 with a TABLO-generated program produced using any of the low memory options (LMC,ECS,SCS,UCS) from the Code stage of TABLO.

Nor are TABLO-like statements 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.

6.7 Coefficients May Not Be Fully Initialised

When write statements are carried out, or the values of coefficients are used in Equations, the values in the coefficient depend on the previous read and formula statements. Note that TABLO-generated programs and GEMSIM do not set all values of a coefficient to zero when you declare that coefficient. This means that some values in a Coefficient may not be initialised before they are used, as the example below makes clear.

Full details about the way TABLO handles this issue can be found in section 4.10.4 of GPD-2.

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 ;
```

TABLO will not even warn that the DVCOMIN("c3",i) values are not initialised.[50] So, when the Write statement is carried out, the DVCOMIN("c3",i) values will not be really determined. The relevant memory in the computer will contain random (that is, unpredictable values).

It is your responsibility to watch out for this sort of problem. If you realise that you are only initialising part of a Coefficient, you can make sure you initialise it all by putting a suitable formula just after you declare it. In the example above, the statement

```
Formula (All,c,COM)(All,i,IND)  DVCOMIN(c,i) = 0 ;
```

(before the Read statements) would do the job.

[50] This is because there are two partial reads into DVCOMIN – see section 4.10.4 of GPD-2.

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 2.11.3 of GPD-1.

7.1 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> ;

where <method_name> can be Johansen, Euler, midpoint or Gragg.

For example method = Euler ;

Euler's method has been introduced in section 2.11.3 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. If you are using Gragg or the midpoint method, the number of steps must be all odd or all even. If you specify just one step number, a single multi-step simulation is done. If you specify two or three, extrapolation is done also.

To obtain information about the convergence of the multi-step methods, if you are extrapolating from 2 or 3 multistep 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. See section 2.11.3 of GPD-1 for an example of extrapolation.

See also section 7.3 for details about subintervals:

subintervals = <number_of_subintervals> ;

7.2 Extrapolation Accuracy Summaries for Data and for Levels Variables

7.2.1 For Data

When extrapolating, it is possible to give estimates of accuracy for each value on each updated data file in the same way as GEMPACK gives such estimates 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 if 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.

The software uses these summaries as the basis for its user-specified accuracy option – see section 7.4.

7.2.2 For Levels 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 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 2.11.3 of GPD-1) shows

```
p_Y  1   6.00000    5.94286    5.91412    5.88527     CX  4 L5
```

The "L5" at the end of this line means that you can be confident of 5 figures accuracy in the level of income Y (and 4 figures in its percentage change p_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 number of figures of accuracy for the levels variables can be used as the basis for user-specified accuracy – see section 7.4.

7.2.3 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. The Extrapolation Accuracy file contains a brief annotation (for example, "EMA 6") about each solution. The Extrapolation Accuracy Summary summarises these.

Note that, if you are extrapolating from only 2 multi-step simulations, no annotations 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.

7.2.3.1 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.[51] This is used in determining the two annotations

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).

Annotations "ERA" and "XRA" never appear unless you have such a statement in your Command file (or select option **RQF** – see section 14.2).

[51] Alternatively you can use option RQF – see section 14.2.

7.2.3.2 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 sent to the terminal. 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.[52]

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 then summarises the XAC-retained variables.)

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 5.8a 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 5.8b 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 but one advantage is that you get an Extrapolation Accuracy Summary (see section 2.11.3 of GPD-1) 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 do this, you can use the you can use the Command file statement[53]

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.

---

[52] 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).

[53] Alternatively select the option

SSI Several subintervals; extrap after each

after which you will be asked for the number of subintervals.

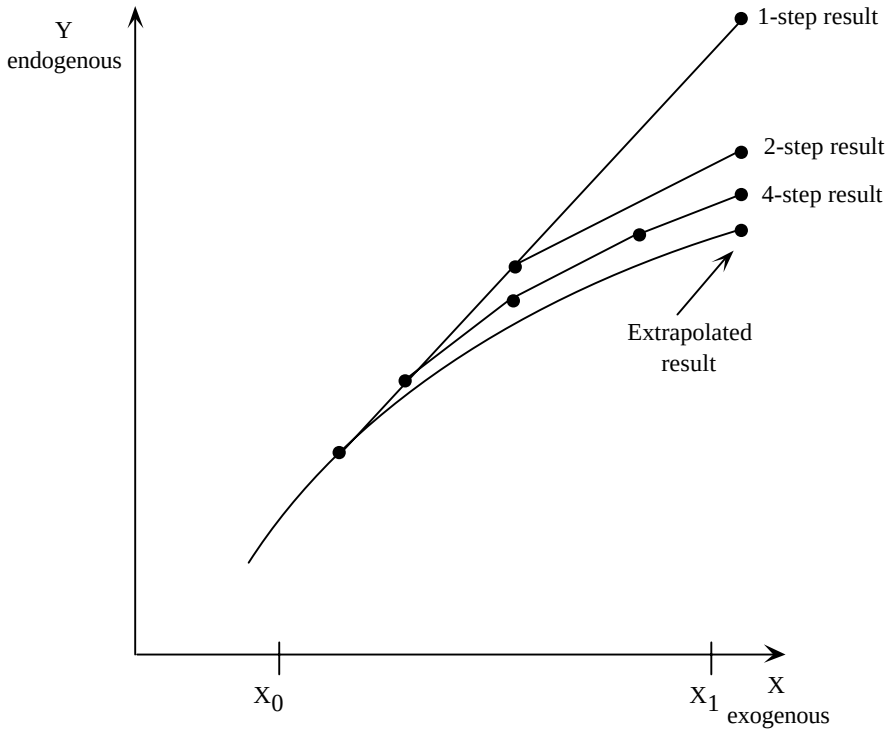


Figure 5.8a: One Subinterval

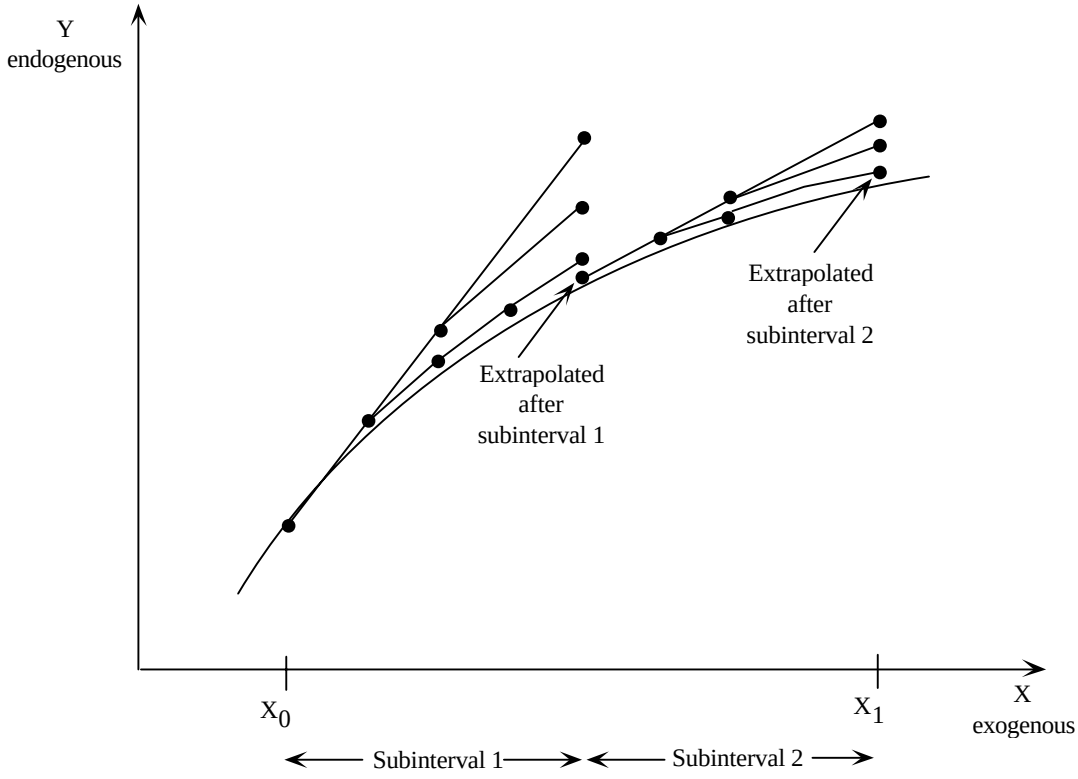


Figure 5.8b: Two Subintervals

An Extrapolation Accuracy Summary is output to the terminal after each subinterval. If you are also creating an Extrapolation Accuracy file, 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 Advantages of Using Several Subintervals

One advantage of using several subintervals in this way is that, as indicated above, you may get early warning of unsatisfactory accuracy.

A second advantage is that, for a given total number of steps (and hence CPU time), you may often get significantly **better accuracy** by using several subintervals. For example, if you have a highly nonlinear simulation requiring large numbers of steps to converge, instead of trying 50,100,150-step calculations, you may get considerably better accuracy from taking 10 subintervals, each with 6,8,10-steps. [In a GTAP simulation 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.]

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 50,100,150-step calculation can have each one far from this path towards the end). [Figures 5.8a and 5.8b above make this moderately clear since, in Figure 5.8a, even if there are 50 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.

7.4 User-specified Accuracy for Simulations

You can guess in advance how many steps to take and how many subintervals (see section 7.3 above) to use to obtain the accuracy you required. Alternatively, 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[54].

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.

```
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 ;
```

[54] This feature was new for Release 5.2 of GEMPACK.

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 7.2.1 above for extrapolation accuracy summaries for the 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> ;
```

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 **accuracy criterion** can be either **data**, **solution** or **both**. When it is data, the Extrapolation Accuracy Summary for all data files (see section 7.2.1 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 7.2.2 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).

When used in conjunction with range restrictions in your TABLO Input file (see section 6.4), simulations carried out with user-specified accuracy ensure that the coefficients never stray outside the specified range.

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 of each subinterval, you can achieve this by selecting just those variables as the XAC-retained ones. 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), with Newton’s method (see section 7.5 below), with the SUP and SSL statements described in section

7.6 and cannot be used in TABLO-generated programs which have been written using any of the low-memory options LMC,ECS,UCS,SCS from the Code stage of TABLO (see section 5.1.2 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 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

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.[55] 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.

---

[55] The defaults are "WARN" if user-specified accuracy is not being used – see section 6.4.4.

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 (or mixed levels/linear) models in certain circumstances.[56] 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).
2. When you run GEMSIM or the TABLO-generated program (after compiling and linking as usual), 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,[57] 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.

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

[56] Newton's method was introduced in Release 5.2.

[57] Alternatively, select option NWT from the initial options screen.

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 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.[58] (Put "simulation = no ;" and "nwt = yes ;" in your Command file to see this information.)

Levels models are available for Stylized Johansen, Miniature ORANI, ORANIF.

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

[58] This was new for Release 7.0.

converge to the “wrong” solution in these cases (that is, to a solution which is not a solution of the underlying levels model).

7.6 Options for Saving Solution and Updated Data Files

7.6.1 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.

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 of GPD-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. 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 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).

7.6.2 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.[59] 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 4.11.7 in GPD-2) in it will be written to this file.

---

[59] Alternatively, select option **SUI** from the opening options screen (see chapter 14). [This is equivalent to the statement “sui = yes ;” in a Command 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.

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 introduce Solution Coefficient (SLC) files.

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 of .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.

  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.

- For Release 7.0 for GEMSIM, TABLO-generated programs as well as SAGEM, your Solution file can now contain subtotals using 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.

- If your TABLO Input file includes levels variables or reports levels variables using the 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 the user in a form that can be read or printed.

- The variable names and set and element information, # labelling information for variables. This can then appear on Print files (PI5) produced by GEMPIE and other output files created by SLTOHT. This information is also used 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.
- The verbal description.
- The closure (which can be recovered using the program SEENV).
- 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 (GEMSIM and TABLO-generated programs only, see chapter 11).
- Individual column solutions (SAGEM only) when the simulation has solved for individual column results – see section 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

- 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 3.9.4 of GPD-1 – 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.[60] 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).

[60] 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** value of all Coefficients defined in the TABLO Input file.[61] 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.

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 Helpfile.

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 will use these to alert the user if the files were not produced at the same time.

A new statement in Command files for GEMSIM and TABLO-generated programs is

SLC file = YES|no ; ! YES is the default.

An SLC file cannot be created by GEMSIM or a TABLO-generated program if an existing Equations file is being used.

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.

[61] SLC files were introduced in Release 7.0.

CHAPTER 9

9. Equations Files and Base Coefficient Values (BCV) Files

Equations files contain numerical versions of the linearized equations of the model. They may be created during a simulation.

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.

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 2.11.1 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.[62]

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 13.1.2 of GPD-4). You will need to create an Equations file if you wish to use SAGEM to carry out Johansen simulations (see section 2.10 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 and save it, using the Command

```
equations file = <file_name> ;
```

The in SAGEM you can use it with the Command

```
use equations file = … ;
```

.

Further details about SAGEM are given in chapter 10.

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 2.11.1 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

---

[62] 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.

$$\mathbf{A}\,\mathbf{z}_1 = -\,\mathbf{D}\,\mathbf{z}_2$$

where $\mathbf{z}_1$ and $\mathbf{z}_2$ are respectively the vectors of endogenous and exogenous variables (see equation (2) in section 2.11.2 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 2.11.1 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 head directly to the equation (*) above on step 1 of a multi-step calculation. It is also what normally happens at steps 2,3... of a multi-step calculation.[63]

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 distinguish between entries of C corresponding to exogenous and endogenous variables, and 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. Thus, in this regard 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 section 5.2), 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, perhaps via a "use LU file … ;" statement in a Command file. This will be allowed if the

[63] TABLO-generated programs written using TABLO Code option SCS do not do this.

model name and version number on the LU file agree with those on the Equations file being used.[64] 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.[65]

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 (the one after Release 7.0).

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.

---

[64] 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.

[65] 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.

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.

10.1 The Solution Matrix

As explained in section 2.10 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 we have seen in section 2.11.2 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).

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 2.10.3 in section 2.10.3 of GPD-1 are the X matrix, of size 27 x 2, for the simulation carried out there.)

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 syntax to retain individual results is:

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

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 2.10.3 in section 2.10.3 of GPD-1 is the T vector for the simulation carried out there.) The Command file syntax to retain cumulative results is:

```
cumulatively-retained endogenous  <list> ;
```

10.1.3 Subtotal Solutions

When running 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 10.1.5 below.

```
subtotal <list>. = <description> ;
```

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 Printing 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 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 X 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.[66] 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 10.2.4, we give more general information relevant to subtotals.

---

[66] 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.

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 and 0, 1 or 2 of the subtotal solutions on the Solution file.

In this first method, the two subtotal solutions are permanently stored on the Solution file, where they can be easily printed via GEMPIE whenever required, simply by referring to their subtotal number. This method is suitable when you have a (small) number of subtotals that you want to refer to often.

10.2.2 Second Method using GEMPIE

Alternatively, you may run SAGEM and just store the 12 individual columns (but no cumulative total or subtotals) on the Solution file.

```
Individually-retained exogenous %all ;
  cumulatively-retained endogenous %none ;
```

You can still obtain the same subtotal solutions by running GEMPIE and specifying two subtotals to be printed, the first summing over the 10 price shock columns and the second over the w and c shocks.

In the second method, the subtotals are not stored on the Solution file but must be defined when running GEMPIE. 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 are to be added up to produce the subtotals column.)

Whenever you specify a new subtotal you will be asked to supply a short description (no more than one line of text) to remind you and others what the subtotal shows.

10.2.4 Printing Subtotals or Individual Column Results using GEMPIE

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 – see chapter 7 of GPD-4.

You can use ViewSOL to access totals and subtotals results produced using SAGEM, but you cannot access individual column results via ViewSOL.

10.3 No Individual 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, 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 2.11.3 in section 2.11.3 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 0 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 2.11.3, 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.[67]

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).

[67] 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.

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.[68] 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.

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 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.

[68] This was introduced in Release 7.0.

In the simple example in section below, there are just 2 shocks and two subtotals results (the consequences of each shock separately). For any endogenous variable, the sum of its results 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 2.5 and 2.6 of GPD-1. We will change the shocks to give a 10 percent shocks 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 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 for model

auxiliary files = sj ;

! Data files

file iodata = sj.dat ;
updated file iodata = <cmf>.upd ; ! modern practice (section 2.5)

! Closure

exogenous p_xfac ;
rest endogenous ;

! Solution method information

method = euler ;
steps = 1 2 4 ;

! Simulation part

! solution file = sjlb ; ! Comment out – modern practice (section 8.1)

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;

subtotal p_xfac("labor") = Effect of labor supply change ;
subtotal p_xfac("capital") = Effect of capital supply change;

! End of Command file
```

Run the new Command file in the usual way described in section 2.5 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 supplied with GEMPACK, as described in detail in section 2.3.4 of GPD-8.

A decomposition of multi-step results in the context of an application with ORANI-G is described in section 2.4 of GPD-8. Again you can carry out this application by working through that section in 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. 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 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 these. Now prompts issued by SLTOHT offer access to subtotals (in addition to the cumulative solution) in appropriate circumstances. See also the new option **SEP** shown in SLTOHT's opening menu. See help on SEP for more details.

The programs RWSOL and MKSOL (see section 11.2 of GPD-4) are able to process these subtotals solutions.

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 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.]

A variation on the example in section 11.3 above will make this clear.

Suppose that we apply all of the labor supply shock and then apply all of the capital supply shock (starting from the updated data after the labor supply shock). The relevant contributions of the two shocks to p_Y (income) would be the results of these two simulations. If you use GEMPACK to calculate these, you will see that these give contributions of 5.8853 (labor change) and 7.565 (capital change) to p_Y. Note that these contributions are different from the subtotals results of 6.1105 and 7.7853 when the two shocks are applied together. Note also that these two contributions of 5.8853 and 7.565 are not a decomposition of the cumulative p_Y result of 13.8958 since they do not add to 13.8958. These contributions would be those obtained if the path from the pre-simulation to the post-simulation values firstly held capital supply constant while increasing labor supply by 10 per cent, and then held labor supply constant while increasing capital supply by 20 per cent.

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.

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 3.9 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 MA28 to MA48, 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 – see section 7.3) instead of a large number of steps in a single multistep solution.
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 initially when you are just testing a new model. Wait until you have checked your simulation results for a few simple simulations, and then you can improve the accuracy later.

An introduction to the procedure by which multi-step solutions are calculated is given in section 2.11.3 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 matrix in equation (3) in 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 calculates the so-called **LU decomposition** (see, for example, section 8.1 of Atkinson (1989) or Stewart (1973)) of A. This is a very 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[69].

[69] 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

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.[70]

Accordingly we are confident in recommending that you ignore the Harwell parameter U completely and just use the default value of 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[71]

NIR = yes ; ! No 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 ;
```

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.

---

time taken to calculate the solution but may slightly increase the rounding errors (and hence slightly reduce the accuracy of the results).

[70] 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.

[71] 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.

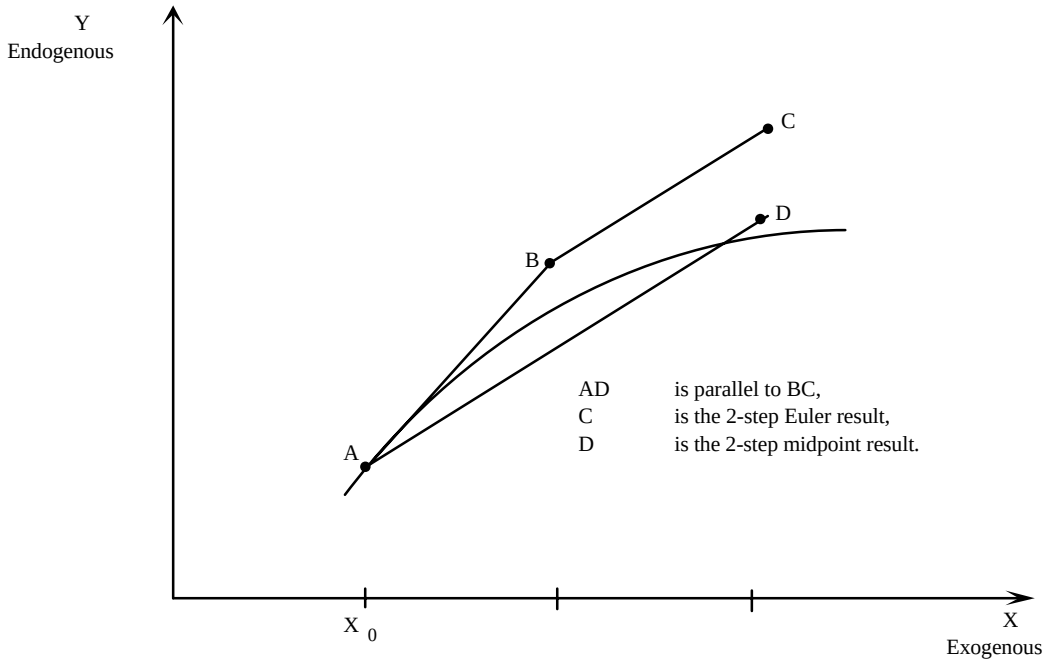
The graphical motivation for Euler's method is shown in Figure 2.11.3 in section 2.11.3 of GPD-1.

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. 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. 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**.)

Our experience is that, provided your simulation is not too nonlinear, Gragg or midpoint will converge significantly faster than Euler and that 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. 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 5.2.7 and 5.2.8 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.

12.3 Reusing Pivots

For large models, the time taken for the LU decomposition (see section 12.1) of the matrix[72] 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.[73]

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 adding the statement[74]

NRP = yes ; ! No Reuse of 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

---

[72] This is the matrix called A(K) in section 6.2. This matrix changes from step to step as the data is progressively updated.

[73] 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.

[74] Alternatively, select option NRP (see section 14.1).

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 simulations 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 the equations based on this data on the Equations file and has stored on the Base Coefficient Values file 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 equations or formulas; instead it reads the values it needs from the Equations and Base Coefficient Values files.
- The LU decomposition does not need to be recalculated, and the solutions for the endogenous variables for this step have already been calculated during step one of the first multi-step calculation.

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.

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, 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 for 2,4,6-step Gragg, the time is likely to be no more than for 2+4+6=12 times that for the Johansen.

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 produced contains all coefficients of the equations as evaluated from the starting data, while the Base Coefficient Values file 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

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.[75]

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 after Release 7.0.

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.

[75] An LU file simply stores the results of the LU decomposition (see section 12.1).

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 3.3.2 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 2.11.1 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 2.2.1a 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 option **IZ1 = yes ;** in your Command file or screen option

IZ1 Ignore zero coefficients in step 1

which tells GEMSIM and TABLO-generated programs to ignore zero coefficients at step 1. Because we have found that keeping zero coefficients in step 1 only speeds up simulations with many steps and only for some models, this option IZ1 is selected by default. If you wish to keep zero coefficients at step 1, you can turn off option IZ1 by putting the statement "**IZ1 = no ;**" in your Command file (or entering option '**-IZ1**') .

At steps 2,3 etc, recording zero coefficients is usually counter-productive (the pivots the program attempts to reuse at step 3 are 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 option **KZ2 = yes ;** in your Command file or screen option

KZ2 Keep zero coefficients in steps 2,3 etc

which tells GEMSIM and TABLO-generated programs to record zero coefficients at steps 2,3 etc. If reuse of pivots is failing, you may like to experiment with option **KZ2** 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.

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 2.11.2 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.[76]

```
%%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 the these warnings about equations not being satisfied very accurately, you should check the equation or equations indicated in the messages.[77]

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.[78] 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.[79]

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.

---

[76] 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).

[77] 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".

[78] 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.

[79] Ending with a fatal error was introduced in Release 6.0. This happens if more than 100 such warnings are given.

```
%%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[80]

NWE = yes ; ! No Warnings about Equations not being satisfied

in your Command file.

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, this does not mean that your simulation is not converging satisfactorily, which is indicated best by the Extrapolation Accuracy Summary or, if in doubt, by the results on the Extrapolation Accuracy file. When you receive messages about variation between passes, you should check this Extrapolation information carefully. If in doubt, perhaps increase the number of steps used in solving the model.

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.

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.[81]

A specific example in which there are increasingly large oscillations is given in section 12.6.4 below.

---

[80] Alternatively, select option NWE (see section 14.1).

[81] Technically, Gragg's method and the midpoint method are only what is called "weakly stable". See, for example, section 6.4 of Atkinson (1989).

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

$$D = 1.1*(-0.18) = -0.198$$

so that the exact simulation result is p_D = -119.8 (that is, a 119.8 per cent decrease in D).

You can, of course, 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 2.2.5 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 2.2 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. And, indeed, you will also see that the Percentage Change differentiation version 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.

But, for this problem, the numerical results obtained from the Change Differentiation rule are increasingly better as the number of steps increases.

General Comments

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 2.2.5 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 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.

CHAPTER 13

13. Compilers and Memory Management

Compilers available for use with GEMPACK are either Fortran 77 or the more recent Fortran 90 compilers. On the PC, the Lahey Fortran compilers **LF90** and **LF95** are in the Fortran 90 class and the Lahey Fortran compiler **F77L3** is in the Fortran 77 class. Compilers on Unix machines can be either Fortran 90 or Fortran 77.

Fortran 90 provides greatly improved memory management compared to Fortran 77. In order to take advantage of these capabilities, the GEMPACK programs have been rewritten using dynamic array allocation available in Fortran 90[82].

There are different Fortran programs (and subroutines) in the source-code version of GEMPACK depending on whether the programs are for a **Fortran 77 or Fortran 90** compiler. Both Fortran 77 and Fortran 90 program source code are supplied on the GEMPACK CD (or disks) but the appropriate programs are installed on your computer differ depending on whether you have a Fortran 77 or Fortran 90 compiler.

The **Executable-image version** of GEMPACK is built using the LF90 compiler, so can be considered as Fortran 90.

The modelling functionality of the Fortran 77 and Fortran 90 versions are identical. The difference is in ease of use because program parameters do not need to be adjusted to accommodate large models when using Fortran 90 GEMPACK programs.

We strongly recommend that you upgrade to Fortran 90 if you are still using a Fortran 77 compiler. Some of the new features in GEMPACK are only available with a Fortran 90 compiler.

In particular, Windows PC users with Lahey F77L3 are encouraged to upgrade to Lahey LF90 or LF95 as soon as possible. F77L3 does not perform well under the later Windows operating systems, notably Windows NT and 2000.

In order to reduce our code maintenance work, we expect **not to support Fortran 77** (including F77L3) in the next release of GEMPACK after Release 7.0.

13.1 Fortran 90 Compilers

13.1.1 Automatic Memory Allocation with Fortran 90

Users with a suitable Fortran 90 compiler (for example, the Lahey compilers LF90 or LF95 on Windows PCs) will find that they do not get messages asking them to increase program parameters.

When used 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.[83]]

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.

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.

---

[82] The Fortran 90 features described in this chapter were new for Release 6.0.

[83] The message will say that the program is unable to allocate sufficient memory.

Apart from the MAXIMUM SIZE change described in section 4.6.2 of GPD-2, the change 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 is the difference.

13.1.2 Preliminary Pass

When GEMSIM or a Fortran 90 TABLO-generated program run, they now 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).]

This it true for both Fortran 77 and Fortran 90 versions of GEMSIM.

In particular, the Fortran 77 version of GEMSIM now uses memory more efficiently than the Release 5.2 version of GEMSIM (since the latter left room for maximum sizes even if the actual sizes were smaller).

13.1.3 Repetitions if Memory Allocation is Too Small

With the Fortran 90 compiler, 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).

However, if you wish to help the program out, you can put one of the statements

```
start with MMNZ|MMNZ1 = <integer value> ;
          ! Sets initial memory allocation
          ! for nonzeros in Fortran 90 GEMSIM or TABLO-generated programs
          ! or SAGEM for Fortran 90
          ! See section 13.1.3..
```

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. These statements have no effect if used with Fortran 77 GEMSIM or TABLO-generated programs.

Note that GEMSIM and TABLO-generated programs can also efficiently increase the size of MMNZ,MMNZ1 if necessary during any step of a multi-step calculation.

SAGEM may also read the Equations file a couple of times before it allocates enough memory for the LU decomposition.

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.

13.1.4 Long File Names Available on Windows PCs with Fortran 90

It is possible to use **long file names and directory names** on Windows PCs with LF90 or LF95. See section 4.9.2 of GPD-1 for details.

13.1.5 Reporting Memory Used by TABLO-generated Programs and GEMSIM

Fortran 90 TABLO-generated programs give reports as to how much memory is required (or used). GEMSIM (Fortran 90 or Fortran 77) does the same.[84]

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[85] for other memory and for memory required for the code itself.

13.1.6 Choice of Fortran 77 or Fortran 90 TABLO-generated Programs

Any version of TABLO can write output for GEMSIM or can write a TABLO-generated program.

The TABLO-generated program can be either a Fortran 77 or a Fortran 90 one;

- the "Fortran 77" program has parameters which may need to be increased[86] while
- the "Fortran 90" program does its own memory management.

If you have a Fortran 90 compiler, you would normally (probably always) choose to write a Fortran 90 TABLO-generated program. If you have a Fortran 77 compiler, you must write a Fortran 77 TABLO-generated program since your compiler cannot compile the Fortran 90 alternative.

[The option **W77** at the CODE stage of TABLO allows any user to write either of these types of TABLO-generated program; if "W77" is selected, TABLO writes the Fortran 77 version while if "W77" is not selected TABLO writes the Fortran 90 version. Appropriate defaults can be set for either compiler.]

If you are writing output for GEMSIM, the output is the same whichever compiler you have.

---

[84] Memory reports were introduced in Release 6.0-001.

[85] For a large model such as MONASH or GTEM, you may have to increase the 4-5 Mbytes to 10 Mbytes.

[86] For example, you may need to increase the value of MMNZ or MMNZ1 (the number of nonzeros in the matrix to be solved). Using Fortran 90, the program realises that this parameter is too small, increases it and tries again until successful (unless it runs out of memory on your computer), whereas under Fortran 77 the program just stops with a message to increase MMNZ, which is particularly annoying in the case of a long multistep simulation.

13.2 Fortran 77 Compilers

If you are using a Fortran 90 compiler, you can ignore all of this section.

13.2.1 No Long File Names or Spaces with F77L3

The Lahey Fortran compiler F77L3 can not handle more than the original 8.3 format (8 character plus a full stop and a 3 character suffix).

You can not use directory or files names containing a space with F77L3.

See section 4.9.3 of GPD-1 for complete details about allowed file and directory names under F77L3.

Some Unix Fortran 77 compilers may be able to handle long filenames or filenames with embedded spaces. Consult your computer manager to find out.

13.2.2 Increasing Parameters in Fortran 77 Main Programs

This section only applies if you are using a Source-code version of GEMPACK in conjunction with a Fortran 77 compiler. It does not apply if you are using a Source-code version of GEMPACK with a Fortran 90 compiler, or if you are using an Executable-image version of GEMPACK on a PC. It does not apply if you are using the Demonstration Version of GEMPACK.

When you run one of the programs, you may receive a message such as that shown below saying that the value of one of the program parameters must be increased.

```
** STOP because must increase the parameter 'MMTVEC'
   from its current value of 170 to at least 193.

  (ERROR RETURN FROM ROUTINE: INCPAR)
  (ERROR RETURN FROM ROUTINE: SAGEM)
```

This message indicates that some aspect of your model has become too large for the program you were running (as this program is currently configured). Provided you have a source-code licence for GEMPACK[87] and there is enough memory on your computer, it is easy to reconfigure the program. All you have to do is edit the source code of the main program in question, changing the relevant value, and then recompile and relink the program.

For the example above, edit the source code of SAGEM (usually called SAGEM.FOR), looking for the line (or part of line) of the form

```
PARAMETER(MMTVEC=170)
```

and change the '170' to '193' or a larger value. (If the message says to increase the parameter by at least one, you should increase it by more since the program probably cannot tell exactly how much it needs to be increased.) To recompile and relink the program, consult your machine-specific GEMPACK documentation (which is GPD-6 on Windows PCs). [On multi-user systems, your GEMPACK Manager may have to do the editing, recompiling and relinking.]

Note that the programs TABLO and GEMSIM are different from the others in this respect. The parameters are set in the TABLO and GEMSIM INCLUDE files (rather than in the source of TABLO or GEMSIM themselves). For details, see section 2.4 of GPD-2 for TABLO and section 13.3.2 below for GEMSIM. Recompiling and relinking TABLO or GEMSIM is more complicated than for the other programs; again consult your machine-specific documentation (which is GPD-6 on Windows PCs) and/or your GEMPACK Manager.

The procedure for increasing parameters in a TABLO-generated program is described in section 13.3.

[87] If you only have executable images of the GEMPACK programs, you will not be able to reconfigure the programs. Either you must reduce the size of the relevant part of your model, or else upgrade to a source-code licence for GEMPACK.

13.3 Increasing Parameters for TABLO-generated Programs and GEMSIM

This section is relevant if you are using a Fortran 77 compiler. It is not relevant if you are using a Fortran 90 compiler unless you select TABLO code option **W77**.

13.3.1 Parameters MMNZ and MMNZ1[88]

If you are using the MA48 routines (see section 12.1), MMNZ1 should be equal to MMNZ. But if you are using the MA28 routines, MMNZ1 probably needs to be larger (perhaps 20 per cent larger) than MMNZ. TABLO-generated programs are written with MMNZ1 equal to MMNZ (since the MA48 routines are now the default).

You may get messages saying that MMNZ and/or MMNZ1 need to be increased. If you are running a TABLO-generated program you can increase the parameter either by editing the TABLO-generated program or else by rerunning TABLO and selecting **option CDM** at the code stage.

Option CDM has a new possible response '**e**' which you can use when you want to specify equal values for MMNZ and MMNZ1.

13.3.2 Increasing Parameter Values in GEMSIM

There are many different program parameters whose values you may need to increase. These affect the maximum size model that can be handled by GEMSIM. (For example, MMNCS sets an upper bound on the number of COEFFICIENT declarations that can be handled currently. If your model has more, you will need to increase the value of MMNCS.)[89]

All of the relevant parameter values have their values set in the two INCLUDE files GSINC and GSINC2 associated with GEMSIM.

Note that, on some machines (for example, Unix machines or VAX/VMS), you may not be able to make these changes yourself, but may have to ask your GEMPACK Manager to make them for you.

To help you find which INCLUDE file to edit, we list below the parameters whose values you are most likely to have to increase, showing which file they are defined in.

FILE	Relevant Parameters in it
GSINC	DCSSTN, DEQSTN, DICMEM, DPCMEM, DRCMEM, DSMRES, DSSMP, DUCMEM, DVCSTN, MCO, MMGSIN, MMGSIC, MMGSIS, MMGSLG, MMGSOP, MMGSRC, MMGSRS, MMNCS, MMNEQ, MMNSS, MMNST, MMNVC, MMNZ, MMNZ1
GSINC2	DBVCSN, DCSRIG, DCSMVC, DCVCSN, DIUX, DIWA, DRCOW, DRUX, DSIMRW, DWA, MBNCOL, MMCM, MMCMCH, MMCOMP, MMEL, MMLIST, MMNCOL, MMNFL, MMNIG, MMNRD, MMNROW, MMNTR, MMNWR, MMSBT, MMSHK

(Don't change the value of any of these parameters unless you receive a message from GEMSIM suggesting you do so.)

If necessary, consult your machine-specific documentation (which is GPD-6 on Windows PCs) to find where the source code for the INCLUDE files is stored on disk (and what their suffixes are).

---

[88] Parameters MMNZ and MMNZ1 have a slightly different meaning in Release 5.2 and later than they did in Release 5.1. With Release 5.2 and later, MMNZ usually needs to be twice as large as the number of nonzero entries in the Equations file (which is about twice as large as in Release 5.1).

[89] If you only have executable images of the GEMPACK programs, you will not be able to reconfigure GEMSIM. Either you must reduce the size of the relevant part of your model, or else upgrade to a source-code licence for GEMPACK.

Once you have changed the appropriate parameter value (by editing the INCLUDE file in which its value is set), you must then recompile and relink GEMSIM. Again, consult your machine-specific documentation to see how to do this.

13.3.3 Increasing Parameter Values for a TABLO-generated Programs

There are four groups of parameters whose values you may need to increase.

- **MMCM** and **MMCMCH** which relate to memory for storing statements on Command files.
- **MMSBT** which is the maximum number of subtotal solutions you can calculate.
- **MMNFL, MMNST, MMNSS, MMNTR, MMNWR, MMNIG, MMEL, DCSRIG, DSSMP** which relate to memory for "extra" TABLO-like statements on Command files (see section 6.6). [For example, if you have more than about 20 "xwrite" statements in your Command file, you may be told that you have to increase the value of program parameter MMNWR.]
- **MMNZ, MMNZ1, MMSHK** and **MMLIST** which relate to simulations.

If you need to increase any of the parameters in the first three groups above, proceed as follows.

- First edit the source code of the TABLO-generated program (the suffix is usually .FOR on Windows PCs or .f on Unix machines) to change the relevant value(s). [The values of these parameters are set near the start of the file. For example, to increase MMCM, search for "MMCM=".][90]
- Then recompile and relink the program (as described in Step 1(b) in section 2.6.1 of GPD-1). [On most machines, use the "LTG" command to compile and link TABLO-generated programs.]

You should never have to increase parameters other than those in the first three groups above in a TABLO-generated program which does data manipulation only (that is, one with no EQUATIONs).

Below we tell you how to increase any of the parameters in the fourth group above. We begin with TABLO-generated programs capable of carrying out multi-step simulations in most detail. TABLO-generated programs which just create the Equations file (but cannot carry out multi-step simulations) are treated at the end of this section.

TABLO-generated Programs Able to Carry Out Multi-step Simulations

There are four program parameters whose values you may need to increase. These allocate the amount of memory available to the program for some part of its processing. The parameters are:

- **MMNZ** which determines the amount of memory available for storing the condensed Equations Matrix C of the linear equations Cz=0. For the program to succeed, MMNZ must be at least twice as large as the number of nonzeros in C at each step. Ideally MMNZ should be somewhat larger to give the Harwell routines a little "elbow room" for sorting the entries of C.
- **MMNZ1** which determines the amount of memory available for LU decomposing the LHS matrix A (in the equations Ay=b). For a simulation to succeed, MMNZ1 must be larger than twice the number of nonzeros in the LU decomposition of A. (MMNZ1 must never be less than MMNZ.)

If you are using the MA48 routines (see section 12.1), MMNZ1 should be equal to MMNZ. But if you are using the MA28 routines, MMNZ1 probably needs to be larger (perhaps 20 per cent larger) than MMNZ. TABLO-generated programs are written with MMNZ1 equal to MMNZ (since the MA48 routines are now the default).

- **MMSHK** which determines the amount of memory available for storing the numerical shocks. For a simulation to succeed, MMSHK must be at least as large as the number of components shocked.

---

[90] If you have difficulty editing the TABLO-generated program, you can change the default values of the parameters in the first two groups as follows. However, this sort of procedure is not available for increasing the parameters in the third group (MMNFL etc). First edit the TABLO Include file TBCDDEC (see section 2.4 above) to change the values of the program parameters DFMCM and DFMCMC; the values of MMCM and MMCMCH are taken directly from these. For MMSBT, change the program parameter DEFMSB. Then recompile and relink TABLO, as described in section 2.4 above, and finally re-run TABLO. Note also that the parameters in these three groups cannot be increased using TABLO Code option CDM (see later in this section).

- **MMLIST** which determines the amount of memory available for storing the program's representation of the various sets of variables (such as the sets of endogenous, or shocked, or cumulatively-retained endogenous variables). The precise role of MMLIST is a little difficult to explain or understand. For a simulation to succeed, for each of these sets, MMLIST must be as large as the sum of the numbers of components of variables which have only some of their components (not zero or all) in the set.

TABLO attempts to make intelligent decisions about the values of these and could usually set values that would guarantee success in all cases. However, to do so would often take considerably more memory than would be needed in most cases. Accordingly, the approach settled on by the designers of the code is to allow user intervention (as described below) to set or influence the values of these. In this way you can intervene to ensure that appropriate values are chosen. This usually shouldn't be necessary but may occasionally be required, especially if you are having difficulty carrying out a given simulation in the available memory.

You can influence or set these values in two ways.

- By selecting option **CDM** at the start of the Code stage of TABLO.
- By editing (via a text editor) the source code of the TABLO-generated program (after TABLO has written it). [The source of a TABLO-generated program is usually in a file with suffix '.for' or '.f'.] After editing, you will need to compile and link the program again (as described in Step 1(b) in section 2.6.1 of GPD-1).

Note that, because TABLO-generated programs for large models are very large files (they can be tens of thousand of lines long), you may not have access to a text editor capable of editing them.[91] But using CDM should always be an option. If want to use CDM and already using a Stored-input file to condense your model, you may have some difficulty anticipating the prompts following your CDM response. In this case, we suggest that you either use option **ASI** (see section 4.3 of GPD-1) or experiment with CDM by running TABLO interactively without doing condensation.

Occasionally you may wish to **decrease** the value of one of the four program parameters because you need to reduce the amount of memory the program requires. If so, you should not edit the TABLO-generated program since this may not reduce the memory required (because of the way the program is written). You should always make the reduction via option CDM.

If a run of a TABLO-generated program has failed because the value of one of these parameters is too small, the TABLO-generated program will usually give you information as to how large it needs to be. However, if the program tells you to increase a parameter by one, you should always increase it by more than this. Then either re-run TABLO and alter the value via Code option **CDM** (see below) or edit the file directly. In the latter case, look for a line in the source code containing the relevant value. For example, to change the value of MMNZ, search for the line containing "MMNZ=". (The value will follow the equals sign and the line should begin with the word "PARAMETER" as in

```
PARAMETER(MMNZ=12345,MMNZ1=23456)
```

which sets the values of MMNZ and MMNZ1.)

MMNZ and MMNZ1 After selecting Code option **CDM** of TABLO, you are given four options relating to MMNZ and MMNZ1. Your possible responses are:

- '**e**'. Here you specify the value of MMNZ and MMNZ1 is given the same value. We recommend that you follow this route if you are using MA48.
- '**s**' Here you specify separate values for MMZ and MMNZ1. We recommend that you follow this route if you are using MA28.
- '**u**' This means you do now want to alter the program's values for MMNZ or MMNZ1.
- '**c**' This means that you want to change the value of DEFMNZ which TABLO uses to set the values of MMNZ and MMNZ1. Because the consequences of this are difficult to predict (or explain), we do not recommend this route.

MMSHK This is set equal to the value of a variable called DEFMSH by TABLO. TABLO starts with a default value for DEFMSH. You can change the value of DEFMSH (and hence of MMSHK) via the CDM

---

[91] With care, it is possible to use a word processor such as Microsoft Word to edit large text files, including TABLO-generated programs. Be sure to save the modified file as a text file. Also be sure to make the page sufficiently wide (and the font sufficiently small) that the lines do not wrap.

option. (However MMSHK never needs to exceed the total number of exogenous variables in your condensed system, and TABLO never sets MMSHK higher than this, even if you put DEFMSH above this value.)

Note that DEFMSH may be zero; if so, DEFMSH is treated as if it were infinity and then MMSHK is set equal to TABLO's best estimate of the number of exogenous variables. (TABLO may need to use a larger value than the actual number of exogenous variables if some of the set sizes are not known precisely until run time.)

MMLIST This is set equal to the value of a variable called DEFMLS by TABLO. TABLO starts with a default value for DEFMLS. You can change the value of DEFMLS (and hence of MMLIST) via the CDM option. Note that DEFMLS must never be set to zero.

The **default** values of DEFMNZ, DEFMSH and DEFMLS (see above) are stored on the TABLO Include file TBCDDEC. If you find that you frequently need to increase values via the **CDM** option, you may like to change the value of one or more of these defaults by editing this Include file and then rebuilding the executable image of TABLO (following the procedure described in section 2.4 of GPD-2).

Other TABLO-generated Programs

For a program which creates the Equations file for your model (but cannot carry out multi-step simulations), only the value of MMNZ is relevant. The procedure for setting and/or influencing it is essentially as above for MMNZ except that, since no "elbow room" is required, 1.2 times MAXTNZ is replaced by just MAXTNZ.

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.[92]

```
                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
NSC Don't save BCV file -->IZ1 Ignore zero coefficients in step 1
NSE Don't save Eq file     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 (though you can, of course, turn it off).

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.

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; see chapter 18 for the syntax of these statements.

---

[92] 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.

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)

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.

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 7.2.3.

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 7.6.2 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 file)

NDS Do no displays ("NDS = yes ;" in Command file)

NWR Do no writes ("NWR = yes ;" in Command file)

NUD Do no final updates ("NUD = yes ;" in Command file)

NAS Do no assertions ("assertions = no ;" in Command file)

NSM Don't do simulation ("simulation = no ;" in Command file)

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 files. 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 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, BPR, LOG, STI, SIF and ASI (which have been described in chapter 5). For SAGEM, these are CPU, CMF, NSM, NIR, NWE and KZC. These extra options will be 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, **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 GEMPACK **Command file**. This is a text file which consists of a sequence of brief commands to give details such as file names, closure and shocks, and a verbal description of the simulation. Select option

CMF Input from a GEMPACK Command file

and give the name of the Command file. Further details are given in section 2.10.2 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, 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 solved – see section 12.6.1 for details. Option

NWE Don't warn how well equations solved ("NWE = yes ;" in Command files)

will suppresses 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, 15.3, 15.4 and 15.5 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 15.2.

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 section 13.2.2 for information about increasing parameters if using a Fortran 77 compiler.

See also the **Frequently Asked Questions** section of the GEMPACK Web site at address:

http://www.monash.edu.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[93] **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), or
- of **zeros** in your data base, or
- that at least one 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 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.

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

[93] 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 2.11.2 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. 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 is 40 less than it should be, 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 is 38 too few, 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[94].
4. Consider switching the values of options IZ1 and KZ2, as discussed 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 skeptical 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.

[94] 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.

15.2 Finding Where an Arithmetic Error Occurs and Rectifying It

Here we are discussing

- divide-by-zero errors (see section 15.3),
- errors arising because of invalid values in powers or in functions such as LOG (see section 15.4), and
- other arithmetic problems such as overflow (see section 15.5),

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. 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 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.

15.3 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 4.9).

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 15.2.

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).

15.4 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 15.2 for advice on how to determine where the error occurs and how to rectify it.

15.5 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 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).

15.5.1 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. Systematic Sensitivity Analysis and Quotas

Procedures for doing **systematic sensitivity analysis** and for handling inequalities including modelling of explicit **import and export volume quotas** in conjunction with any model solved using GEMPACK have been developed and automated. These two sets of procedures (which are not related) are described in this chapter.

16.1 Systematic Sensitivity Analysis

Results of simulations often hinge critically on values of key exogenous inputs (the values of the parameters of the model and/or the shocks applied). Computational burden has, in the past, hindered systematic investigation of the impacts of variations in these inputs. Arndt and Pearson (1998) describe practical methods for conducting systematic sensitivity analysis [**SSA**] for any model solved using GEMPACK.

In many cases of interest, the model only needs to be solved 2N times if N inputs (parameter values or shocks) are varying. Automated procedures for working out how to vary these inputs and for carrying out the associated solves of the model are given in the software accompanying the paper. Once you set it up (which requires only small changes from the original single simulation), no further intervention is required. The procedure reports estimates of the mean and standard deviation for any endogenous variable in the model.

These methods are automated in RunGEM (see chapter 5 of GPD-8) via its ***Tools*** menu.[95]

The practicality of the methods described is due partly to recent discoveries in the area of numerical quadrature made by Paul Preckel (Purdue University) and his graduate students - see, for example, DeVuyst and Preckel (1997) and Liu (1996). An application of these methods can be found in Arndt and Hertel (1997).

16.2 Explicit Volume Quotas and Other Inequalities

GEMPACK's solution algorithm (including extrapolation) relies on the underlying levels equations being smooth (that is, differentiable). Inequalities such as explicit volume quotas do not appear to fit into this category. However our colleague Mark Horridge [Horridge (1993)] has worked out procedures for working around this difficulty.

The essential ideas behind Horridge's procedure are that

1. these problems can easily be converted to smooth ones for which extrapolation works once we know the post-simulation status of the relevant inequalities (for example, whether particular imports are below or at the quota).
2. an Euler version of the simulation with sufficiently many steps should be able to tell us the post-simulation status of these inequalities.

The general procedure is to

- first run an approximate simulation with sufficiently many Euler steps to find out the post-simulation status of the relevant inequalities,
- then calculate the shocks needed to bring about these,
- finally run an accurate simulation (using extrapolation) modifying the closure and feeding in the shocks just calculated.

---

[95] The automation of these procedures in RunGEM means that the software prepared to accompany Arndt and Pearson (1998) is no longer of interest (except possibly on computers other than Windows PCs).

The results of the accurate simulation are the desired ones.

Data-manipulation TABLO Input files are needed to check the post-simulation status of the relevant inequalities and to calculate the shocks required to achieve these starting from the pre-simulation data. These must also check (usually via ASSERTIONs) that the post-simulation status after the accurate simulation is the same as that after the approximate simulation; if not, the approximate simulation needs to be rerun with more Euler steps.

The closure swaps needed for the accurate simulation can be specified via the method described in section 5.2.4. The data files used are the shocks file prepared by the data-manipulation program which is run after the approximate simulation. [See the technical papers referred to in section 16.2.1 or 16.2.2 for specific examples.]

Note that the range check qualifiers described in section 6.4 and in section 4.5.7 in GPD-2 are not replacements for the techniques described in the sections below. The range qualifiers (together with automatic accuracy) are designed for cases where the underlying equations are differentiable, which is not the case when there are explicit quotas in place.

16.2.1 Explicit Volume Quotas

Bach and Pearson (1996) describes how to model explicit volume import and export quotas in the GTAP model. The techniques described there apply equally well to any model implemented with GEMPACK. They also generalise to other inequalities. [For example, Michael Malakellis (1994) has used these methods in an intertemporal model of the Australian economy to ensure that investment stays non-negative.]

Accompanying Bach and Pearson (1996) are various computer files containing ingredients for examples readers can carry out for themselves. These files can be used as the starting point for those who wish to explicitly model quotas in their own model. They are available from the World-Wide Web at address

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

The quota procedures described are automated to make them easy to use. The facility for specifying closures using data files described in section 5.2.4 above was developed in this context.

16.2.2 Tariff Rate Quotas

These are import tariffs whose rate increases once import volumes exceed a specified level. Elbehri and Pearson (2000) describe how these can be modelled in the GTAP model. Accompanying this paper is a Windows program TRQmate which automates the running of the two associated simulations (an approximate one and then the accurate one) and of the associated checking and shock-generation TABLO-generated programs.

CHAPTER 17

17. New Command File Statements

In this chapter we show all the Command file statements which are new for GEMPACK Release 7.0 or were new for Release 6.0 or Release 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 7.0

SLC file = YES|no ; ! see section 8.4

xtransfer <header> from file ! see section 4.12 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 5.5.5

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.

17.2 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 5.5.4

scale equations = yes|NO ; ! See section 12.6.5

```
start with MMNZ|MMNZ1 = <integer value> ;
       ! Sets initial memory allocation for nonzeros
       ! in Fortran 90 GEMSIM or TABLO-generated programs.
       ! or Fortran 90 SAGEM. See section 13.1.3


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 5.5.5
```

17.3 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 7.6.2 and 14.3
updated formula initial data = <filename> ;  ! see section 7.6.2

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 7.6.1
SUP = all | last | NONE ;   ! see section 7.6.1

exogenous|endogenous <variable> = zero|nonzero|positive|negative
        value on file <file> ;  ! see section 5.2.4
! 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 7.0 of GEMPACK.[96]

Command files were introduced in chapter 2 and also in GPD-1 in sections 2.5.5, 2.5.6, 2.8.1 (Figure 2.8.1) and 2.10.4 (Figure 2.10.4). 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.

[96] The material in this chapter is an updated version of the documentation that was in Chapter 19 of the Release 6.0 edition of GPD-4 and Appendix A of the Release 5.2 edition of GPD-1.

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

steps = <list_of_step_numbers> ; ! There is no default
! 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
! EXAMPLE. accuracy figures = 5 ;

accuracy percent = <pp> ; ! 80 percent is the default
! EXAMPLE. accuracy figures = 90 ;

accuracy criterion = DATA|solution|both ; ! data is the default
! EXAMPLE.. accuracy criterion = both ;

! The next statements are new for Release 6.0 GEMPACK

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
! EXAMPLE. updated terminal data = sjlb_term.upd ;

intermediate file <logical_name> = <actual_name> ;
! For intermediate updated versions.
! 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 is 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**. 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 ;
! new for Release 6.0
! 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.
! EXAMPLE. version = 3 ;

identifier = <identifier> ; ! to specify the model identifier on a new Equations file
! DEFAULT is all blanks
! 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**. 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
! 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)

display file = <file_name> ; ! include suffix

! EXAMPLE. display file = sj.dis ;
! The following is new for Release 6.0 GEMPACK
! default if statement is not present *display file = command file stem + .dis ;*
! provided the Command file ends with ".CMF" (any case allowed)
! see section 0 for details

Extrapolation Accuracy file = yes|NO ; ! default is NO

! Use 'yes' to get an XAC file
! see section 7.1
! 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 section 3.2).

! 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

use Environment|LU file <file_name> ;

! EXAMPLE. use Environment file sjxfac ;
! omit suffix '.en4'
! If 'use Environment file', just the closure is read from it.
! If 'use LU File', the closure AND LU decomposition are read
! from it.

! 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 LU file sjxfac ;
! omit suffix '.lu4'

! If 'take closure from LU File', the closure is read from it, but the LU decomposition is
! calculated from scratch.

! Note that 'take closure from Environment file' has exactly the
! same effect as 'use Environment file'.

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.

! If 'modify closure from LU file', the LU decomposition on the file is not used.

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 ;

save Environment|LU file <file_name> ;

! EXAMPLE. save LU file sj1 ;
! omit suffix '.lu4'

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

! see section 5.2.4
! This is new for Release 5.2 GEMPACK
! 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.)
! 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 5.5.4 - This is new for Release 6.0 GEMPACK.

ashock … ; *! Additional shock statement.* For example “ashock xtax = uniform 10 ;”.
tshock … ; *! Target shock statement.* For example “tshock xtax = uniform 10 ;”.
See section 5.5.5 ! This is new for Release 6.0 ! (not allowed with SAGEM)

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'

! Only has effect if 'Extrapolation Accuracy file = yes ;'

! command is included (see above)

SUBTOTALS

! New for Release 7.0 see section 11.1

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.
! see section 11.1

! EXAMPLE. subtotal px 1-10 15-20 pm 2-4 zz = subtotal 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 4.3.3 in GPD-1.

! The following was new for Release 6.0 GEMPACK
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 - this may change
! write all pre-simulation Coefficient values to the SLC file
! new for Release 7.0 for use with AnalyseGE program
! see section 8.4

! 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.
! Use 'NIR = yes ;' to turn off iterative refinement of solutions.

IZ1 = YES|no ; ! default is YES.[97]
! 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,...

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.

[97] In Release 5.0, the default was NO.

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 7.6.1 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 7.6.1 for details

scale equations = yes|NO ; ! default is NO
! Option new for Release 6.0
! See section 12.6.5 for details

start with MMNZ|MMNZ1 = <integer value> ; ! Sets initial memory allocation for
! nonzeros in Fortran 90 GEMSIM or TABLO-generated programs.
! See section 13.1.3. New for Release 6.0. 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.
! Use 'NEL = yes ;' to suppress element name/number
! labels on row_order or col_order text files.

18.1.2 General Points

1. Input is free-form and case-independent. In particular, statements can extend over more than one line, such as

   exogenous pfac xcom xfacin 1-4 6-19
   yfacin z ;

   It doesn't matter if your command is 'Equations file = ... ' or 'equations FILE = ... '.

2. "|" indicates alternatives.

3. < > shows user selected text; all other text is required, but words in it 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'.)

4. [ ] indicates optional user-selected text.

5. 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.

6. 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) 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,
   Shocks only given to domestic inputs!! Foreign ones given no shock.

7. 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 ;]

8. Specifying all or some components of a variable.

   In the statements above, the notation

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

   covers three possibilities, as described below.

   (a) Components can be indicated by number, for example

   x3 2-12 15-20

   (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.

9. SHOCKS. The components must be specified in increasing order, as in

```
shock v1 3 4 8 10 = 1.1 4.4 3.3 2.4 ;
```

10. Shocks can be input using the repeated value format as described in section 6.2.7 of GPD-4. For example,

```
shock v1 3 4 8 10 12-16 = 3*1.345   4.4  5*-2.1 ;
```

11. To shock some components of a multi-dimensional variable, remember that the first index varies fastest (see section 6.1.3) and specify component numbers accordingly. For example, for a 3x5 variable V, to shock V(2,1), V(3,1), V(2,2),

```
shock v 2 3 5 = 4.5 2.0 1.0 ;
```

12. 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,

```
equations file is SJLB.EQ4,
updated file iodata is SJLBabc.upd
```

See section 2.5.4 for details

18.1.3 Complete Example

```
!_____________________________________________________________________
!
! 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
! 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

save Environment|LU file <file_name> ;
! EXAMPLE. save LU file sj1 ;
! omit suffix '.lu4'

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.)

! 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.

use Environment|LU file <file_name> ;

! EXAMPLE. use Environment file sjxfac ;
! omit suffix '.en4'
! If 'use LU File', the closure AND LU decomposition are read from it.
! If 'use Environment file', just the closure is read from it.

take closure from Environment | LU | Solution file <file_name> ;

! EXAMPLE. take closure from LU file sjxfac ;
! omit suffix '.lu4'
! If 'take closure from LU File', the closure is read from it,
! but the LU decomposition is calculated from scratch.

! Note that 'take closure from Environment file' has exactly the
! same effect as 'use Environment file'.

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.

! If 'modify closure from LU file', the LU decomposition on
! the file is not used.

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 ;

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

cumulatively-retained endogenous <list> ;
! default is: all endogenous variables

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.

! 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 4.3.3 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 Fortran 90 SAGEM .
! See section 13.1.3. New for Release 6.0.

CPU = yes|NO ; ! default is NO.
! Use 'CPU = yes ;' to get CPU times reported.
! (This may not produce meaningful results on some machines)

NIR = yes|NO ; ! default is NO.
! Use 'NIR = yes ;' to turn off iterative refinement of solutions.

KZC = yes|NO ; ! default is NO.
!.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.
! Use 'NWE = yes ;' to suppress any warnings about equations
! not being solved accurately.

scale equations = yes|NO ; ! default is NO
! See section 12.6.5 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 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

Arndt, C. and K.R. Pearson (1998), Implementing Systematic Sensitivity Analysis using GEMPACK, GTAP Technical Paper No.3, April 1998, pp.45+5.

Arndt C. and T.W. Hertel (1997), Revisiting the Fallacy of Free Trade, *Review of International Economics*, Vol 5, Issue 2.

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.

DeVuyst, E.A. and P.V. Preckel (1997), Sensitivity Analysis Revisited: A Quadrature-Based Approach, *Journal of Policy Modelling*, 19(2) pp.175 -185.

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.

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, Draft GTAP Technical Paper, currently under review at Purdue University.

Harrison, W.J., J.M. Horridge and K.R. Pearson (2000), 'Decomposing Simulation Results with Respect to Exogenous Shocks', *Computational Economics* (to appear). [A preliminary version was *Centre of Policy Studies and Impact Project Preliminary Working Paper* No. IP-73, May 1999.]

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.]

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.

Liu, S. (1996), Gaussian Quadrature and Its Applications, PhD Dissertation, Department of Agricultural Economics, Purdue University.

Malakellis, Michael (1994), ORANI-INT: An Intertemporal GE Model of the Australian Economy, PhD Thesis, Centre of Policy Studies and Impact Project, Monash University.

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.

Stewart, G.W. (1973) *Introduction to Matrix Computations*, Academic Press, London.

20. GEMPACK DOCUMENTS[98]

Harrison, W.J. and K.R. Pearson (2000), *An Introduction to GEMPACK*, GEMPACK Document No. 1 **[GPD-1],** Monash University, Clayton, Fifth edition, October 2000.

Harrison, W.J. and K.R.Pearson (2000), *TABLO Reference*, GEMPACK Document No. 2 **[GPD-2]** Monash University, Clayton, Third edition, October 2000.

Harrison, W.J. and K.R.Pearson (2000), *Simulation Reference: GEMSIM, TABLO-generated Programs and SAGEM*, GEMPACK Document No. 3 **[GPD-3]**, Monash University, Clayton, October 2000.

Harrison, W.J. and K.R.Pearson (2000), *Useful GEMPACK Programs*, GEMPACK Document No. 4 **[GPD-4]** Monash University, Clayton, October 2000.

Harrison, W.J. and K.R. Pearson (2000), *Installing and Using the Source-Code Version of GEMPACK on DOS/Windows PCs with Lahey Fortran*, GEMPACK Document No. 6 **[GPD-6],** Monash University, Clayton, Tenth edition, October 2000.

Harrison, W.J. and K.R. Pearson (2000), *Installing and Using the Executable-Image Version of GEMPACK on DOS/Windows PCs*, GEMPACK Document No. 7 **[GPD-7],** Monash University, Clayton, Seventh edition, October 2000.

Harrison, W.J. and K.R. Pearson (2000), *Getting Started with GEMPACK: Hands-on Examples*, GEMPACK Document No. 8 **[GPD-8],** Monash University, Clayton, Second edition, October 2000

[98] 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, 2-7, 18-123

A

Abbreviation
  Command file syntax, 2-9, 18-137
Acceptable Range
  Coefficient, 6-53
Accuracy, 7-61, 12-89
  Different methods, 12-92
  Iterative refinement, 12-90
  Required figures, 7-61
  Reuse of pivots, 12-92
  Subintervals, 7-62
  User-specified, 7-64
  Warnings, 12-96
Actions
  GEMSIM and TABLO-generated program, 6-43
  Options, 14-110
Actual data file
  Connection with logical file, 4-15
Additional shocks, 5-37
Arithmetic errors, 15-118
Arithmetic overflow
  Meaning, 15-118
ashock statements, 5-37
Assertions, 6-44, 6-45, 6-51
Automatic accuracy, 18-125. *See also* User-specified accuracy
Auxiliary files
  Check, 3-13
  GEMSIM, 3-11
  Names, 3-12
  TABLO-generated program, 3-12
Auxiliary Statement, Table files, 3-11, 3-13, 18-128

B

Base Coefficient Values file. *See* BCV file
Base Coefficient Values File, 12-94
BCV file, 9-75, 9-77
  Introduction, 9-75
  Use not supported in next release ?, 9-77
  Using, 9-77

C

C matrix, 10-83
CDM
  TABLO Code option, 13-107
Check
  ASSERTION, 6-51
Check on read options
  CRN,CRW,CRF, 4-21
Checking closure, 5-41
Checking shocks, 5-41
Check-on-read
  defaults, 4-20
Check-on-read statements, 4-20
Choice of variables
  Command file, 5-31
  Interactive, 5-34
  To shock or print, 5-31
Closure, 5-25
  Command file syntax, 18-130
  Different shocks, 10-83
  Meaning, 5-25
  Modifying commands, 5-29
  Saving, 5-28
  Sets, 6-56
  Specifying, 5-27
  Valid, 5-25
  When Processed, 6-49
CMFSTART file, 4-21, 13-102
Coefficient
  Acceptable Range, 6-53
Coefficient initialisation, 6-58
Coefficients
  Values may not be fully initialised, 6-58
  Zero, 12-95
Column order, 6-46, 14-111
Columns
  Solution, 10-79
Command file, 18-123
  <cmf>, 2-6
  Closure, 5-27
  Eliminating syntax errors, 2-10
  Example, 18-139
  Name, 2-5
  SAGEM, 5-33
  SAGEM example, 18-145
  Statements, 17-122
  Variable Components, 2-9
Command file stem, 2-7
Command file syntax
  Abbreviation, 2-9, 18-137
  Automatic accuracy, 18-125
  Case independent, 2-8, 18-137
  Choice of variables, 5-31
  Closure, 18-130
  Files, 18-127
  SAGEM, 18-140
  Shocks, 5-35, 18-132
  TG Programs and GEMSIM, 18-124
  Verbal description, 18-131
Command files, 4-17, 14-109
  <cmf> for file names, 2-6, 2-7
  Display file name, 2-7
  Extra statement, 6-56, 6-57
  Log file name, 2-7
  Minimum subinterval statements, 7-66
  Range test, 6-55
  Solution file name, 2-7
  TABLO-like statements, 6-56, 6-57
Comments, 6-56
  Command file syntax, 2-9, 18-137
Component list
  Shocks, 18-132
Component numbers, 5-30

Convergence, 12-92
CPU time, 2-8, 14-111
  SAGEM, 14-112
Cumulative results, 10-80
Cumulatively-retained
  Choice of, 5-31
Cumulatively-retained endogenous
  Meaning, 10-80

D

D1C
  Display file - 1-dimensional arrays, 4-18
Data
  Extrapolation Accuracy Summary, 7-65
Data file
  Command file syntax, 18-126
Data files
  Intermediate, 4-17
Data manipulation
  Actions, 6-44
Decomposing results, 11-87
  Applications, 11-87
Default filenames, 2-7
Default options
  GEMSIM, TG programs, 14-109
Defaults
  Command file, 2-9, 18-137
Display
  All steps, 14-111
  Command file, 6-56
Display decimals, 4-18
Display file name
  In Command files, 2-7
Display files, 4-18
Display Files
  Options, 4-18
Display length, 4-18
Display width, 4-18
Displays, 6-44, 6-45
Divide by zero Errors, 15-113
DOI
  Display file - "identical" messages, 4-19
DPN
  Display file - new pages?, 4-18
Dynamic models, 5-37

E

Echo activity, 6-46
Elasticity, 10-79
Element names
  Reading, 4-21
Endogenous, 5-25
  Individual, cumulative, 10-80
End-point
  Gragg's method, 12-92
Environment
  Meaning, 5-25
Equations
  Scaling, 12-99
  Solving, 12-89
  Solving them, 12-90
Equations file, 9-75
  Command file syntax, 18-127
  Creation, 6-43
  Introduction, 9-75
  Using with BCV file, 9-77
Equations File, 12-94
Equations files, 9-75
Equations matrix, 9-75
Equations Matrix, 10-83
Equations not satisifed very accurately
  Very serious, 12-96
Equations not solved very accurately, 14-112
Error
  Extra statement, 6-57
Error message, 13-104
Error trapping
  Invalid argument in function, 15-117
  Invalid power, 15-117
Errors, 15-113
  Arithmetic, 15-118
  Overflow, 15-118
Errors in Command file
  Eliminating, 2-10
Euler's method, 6-47, 12-91
Exogenous, 5-25
  Individual, cumulative, 10-80
Extra data file
  Command file syntax, 18-126
Extra statements, 6-56, 6-57
Extra TABLO-like statements, 6-56
  Qualifiers, 6-57
Extrapolation, 6-47
  Subintervals, 7-62
Extrapolation Accuracy file, 7-64
  Variables, 7-62
Extrapolation Accuracy summary, 12-92
Extrapolation Accuracy Summary, 7-61, 7-64, 7-65
  Data, 7-60
  Levels variables, 7-60

F

Fatal Error, 13-104
File
  LU decomposition, 12-94
  Parameter, 4-17
File name
  Long, 13-104
  Spaces, 2-5
File statements in Command files
  Purpose, 4-15
Filename
  Default, 2-7
Flexible style
  Element names, 4-21
Forecasting models, 5-37
FORMULA(INITIAL)
  Save updated values, 7-69, 14-110
Fortran 77
  Not supported after Release 7.0 ?, 13-101
Fortran 90
  Memory management, 13-101

G

GEMPIE
  Choice of variables, 5-31
  Print file, 10-81
GEMSIM
  Command file syntax, 18-123
  Equations file, 12-94

Increasing parameters, 13-105
GEMSIM and TG programs
Option – CR, 4-21
Option – CRF, 4-21
Option – CRN, 4-21
Option – CRW, 4-21
GEMSIM and TG Programs
Command file example, 18-139
GEMSIM Options
Menu, 14-109
GEMSIM Statement, Table files, 3-11
Gragg's method, 6-47
Gragg's method
Number of passes, 12-92
Unsuitable in some cases, 12-97
Graggs's method
Graphical explanation, 12-91
GTEM model, 5-37

H

Harwell Laboratories, 12-89, 12-90
Harwell parameter, 12-89
Harwell subroutine, 12-95

I

Ill-conditioned matrix, 12-96
Increasing Parameters, 13-104
Individual column results, 10-79
GEMPIE, 8-73, 10-81
Not available in multi-step simulation, 10-83
Not available via GEMSIM, 10-83
Not available via TABLO-generated program, 10-83
Not available via ViewSOL, 8-73, 10-81
Individual Column results
Printing, 10-82
Individually-retained, 10-79
Choice of, 5-31
Inequality, 16-119
Input files, 4-16
Intermediate extra data file, 4-22
Intermediate file
Command file syntax, 18-126
Internediate data files, 4-22
Invalid power, 15-117
Iterative refinement, 12-90

J

Johansen simulation, 10-79
Johansen's method, 6-47

K

Keyword
Extra statement, 6-56

L

Levels equations
Newton's method, 7-67
Levels results, 8-73
GEMPIE, 8-72
ViewSOL, 8-72
Levels variable
Acceptable range, 6-53
Zero, 12-92
Lists
Choice of variables, 5-31
Log file name
In Command files, 2-7
Log files, 2-8
Log only, 2-8
Logical file, 4-15
Connection with actual file, 4-15
Long file names, 13-104
LU decomposition, 6-49, 12-89
LU File, 12-94

M

MA28, 12-89
MA48, 12-90
Macro variables, 5-32
Matrix
Equations, 10-83
Singular, 12-96
Memory
Insufficient, 13-101
Memory Allocation
Fortran 90, 13-101
Memory required
by GEMSIM, 13-103
by TABLO-generated program, 13-103
Relating to MMNZ,MMNZ1, 13-103
Midpoint method
Graphical explanation, 12-91
Number of passes, 12-92
Midpoint method
Unsuitable in some cases, 12-97
MIN file. *See* Model Information file
Miniature ORANI
Example, 5-25
MMNZ, 13-105
MMNZ and MMNZ1, 13-102
Memory required, 13-103
MMNZ1, 13-105
Model identifier, 9-76
Model Information file, 3-13
Model name, 9-76
Model version number, 9-76
Modified midpoint method, 12-92
Modifying a closure, 5-29
MONASH model, 5-37
Multi-step simulation, 6-47
Multi-step solution, 12-89
No individual column solutions, 10-83

N

Names of Auxiliary files, 3-12
Newton correction terms, 7-68
Newton variable
$del_newton, 7-67
Newton's Method, 7-67
Nonlinear, 7-62, 12-92
Number of steps, 12-92
Numerical values
Shocks, 5-35
Numerically singular, 12-96, 15-114

O

Options
GEMSIM and TG programs, 14-110
GEMSIM and TG programs - CPU, 14-111
GEMSIM and TG programs – CPU, 2-8
GEMSIM and TG programs - D1C, 14-111
GEMSIM and TG programs - DDC, 14-111
GEMSIM and TG programs - DOI, 14-111
GEMSIM and TG programs - DPL, 14-111
GEMSIM and TG programs - DPN, 14-111
GEMSIM and TG programs - DPW, 14-111
GEMSIM and TG programs - DTO, 14-111
GEMSIM and TG programs - DWS, 14-111
GEMSIM and TG programs – DWS, 6-46
GEMSIM and TG programs - EAA, 14-111
GEMSIM and TG programs – EAA, 6-46
GEMSIM and TG programs - IZ1, 15-114
GEMSIM and TG programs - KZ2, 12-95, 15-114
GEMSIM and TG programs – NAX, 3-11
GEMSIM and TG programs - NDS, 14-110
GEMSIM and TG programs - NEL, 14-111
GEMSIM and TG programs - NEQ, 14-110
GEMSIM and TG programs – NIR, 12-90
GEMSIM and TG programs - NRP, 12-92
GEMSIM and TG programs - NSC, 14-110
GEMSIM and TG programs - NSE, 14-110
GEMSIM and TG programs - NSM, 14-110
GEMSIM and TG programs - NUD, 14-110
GEMSIM and TG programs - NWR, 14-110
GEMSIM and TG programs – RQF, 7-61
GEMSIM and TG programs - SCS, 9-76
GEMSIM and TG programs - SSI, 7-62
GEMSIM and TG Programs - SUI, 7-69, 14-110
GEMSIM and TG programs – SVX, 7-62
GEMSIM and TG programs - TWC, 14-111
GEMSIM and TG programs – TWC, 6-46
GEMSIM and TG programs IZ1, 12-95
GEMSIM and TG Programs M28, 12-90
Help, 14-109
SAGEM - Command file, 18-144
SAGEM CMF, 14-112
SAGEM CPU, 14-112
SAGEM KZC, 14-112
SAGEM NIR, 14-112
SAGEM NSM, 14-112
SAGEM NWE, 14-112
TABLO - ACD, 12-98, 12-99
TABLO - CDM, 13-105
TABLO - NTX, 8-73
TABLO – NWT, 7-67
TABLO - W77, 13-103, 13-105
TABLO Code - CDM, 13-107
Options List
GEMSIM and TGPrograms, 18-134
Options Menu
TABLO-generated programs, 14-109
Order
Reads, formulas, equations, 6-47
Oscillating results, 12-97
Out of memory, 13-101
Output files, 4-16
Outputs from SAGEM, 10-79
Overflow
Meaning, 15-118

P

Parameter
Code, 15-113
Parameter files, 4-17
Parameter values
Decreasing, 13-107
Increasing, 13-105
Parameters
Program, 13-104
Parameters MMNZ and MMNZ1, 13-105
Passes
TABLO-generated program or GEMSIM, 6-47
Percentage change
Large negative, 12-92
Pivots, 12-92
Preliminary pass
Set sizes, 13-102
Prompts
Choice of variables, 5-31

Q

Quotas, 16-119

R

Range check. *See also* Range test
Range checks, 6-44
Range tests, 6-44, 6-45
Reading
Element names, 4-21
Repeated values, 2-9, 5-40, 18-138
Residual correction, 12-90
Reusing pivots, 12-92
May not be beneficial with MA48, 12-93
Rounding error, 12-89
Row order, 6-46, 14-111
Row totals, 10-80
RunDynam, 5-37, 5-42
CMFSTART file, 4-21, 13-102
RunGEM
CMFSTART file, 4-21, 13-102
RunMONASH, 5-37, 5-42
CMFSTART file, 4-21, 13-102

S

SAGEM
Choice of variables, 5-31
Command file example, 18-145
Command file syntax, 18-140
Equations file, 12-94
Johansen simulation, 10-79
Solution file, 10-80
Subtotal, 10-80
SAGEM Options, 14-112
Save Updated values
SUI, 7-69, 14-110
Saving separate multi-step solutions, 7-69
Saving updated data after each multi-step solution, 7-69
Scalar variables, 5-32
Scaling
Equations, 12-99
Scaling equations
Not recommended generally, 12-99

Select from file, 5-36
Sensitivity Analysis, 16-119
Set
  Command file, 6-56
Set element name
  Reading, 6-57
Set elements
  Reading, 4-21
Set/element information
  checking when reading data, 4-19
Setting MMNZ and MMNZ1, 13-102
Several Subintervals, 7-64
Shock files, 5-40
Shocks
  Command file syntax, 18-132
  Component list, 2-9, 18-132
  For one step, 6-50
  Meaning, 5-25
  Multi-dimensional variable, 5-40
  No shock statements, 5-38
  Numerical values, 5-35
  Repeated values, 2-9, 18-138
  Same closure, 10-83
  Select from file, 5-36, 18-132
  Some exogenous, 5-38
  Specifying, 5-35
  Subintervals, 7-62
  When Processed, 6-49
Shocks – Uniform
  Meaning, 5-35
Simulation
  Choice of variables, 5-31
  SAGEM, 10-79
Singular, 15-114
Singular matrix, 15-113
SLC file, 8-74
Software
  SSA, 16-119
Solution
  Extrapolation Accuracy Summary, 7-65
  Matrix, 10-79
  Simulation, 10-79
  Subtotal, 10-80
Solution Accuracy, 12-96
Solution Coefficients (SLC) file, 8-74
Solution file
  Saving multi-step, 7-69
Solution file =
  Recommendation not to include this, 8-71
Solution file name
  In Command files, 2-7
Solution files
  Contents, 10-79
  SAGEM, 10-79
Solution method
  Command file syntax, 7-59, 18-124
  Verbal description, 8-71
Solution time, 12-92, 12-93
  Existing files, 12-94
  Subintervals, 7-62
  Total, 12-93
Solutions
  Saving after each multi-step calculation, 7-69
Solving
  Levels equations, 7-67
Spaces
  File name, 2-5
Sparse matrices, 12-92, 12-95
Sparse Matrices
  MA48, 12-90
Sparse matrix, 12-89
Speed of Solution
  Total time, 12-93
Speed of solution, 12-92, 12-93
  Existing files, 12-94
  Iterative refinement, 12-90
  Subintervals, 7-62
  Zero coefficients, 12-95
SSA, 16-119
SSL statements in Command files, 7-69
Steps
  Command file syntax, 7-59, 18-124
  Number, 12-92
  TABLO-generated program or GEMSIM, 6-47
  Verbal description, 8-71
Steps, passes, 12-92
Structurally singular, 15-114
Subintervals, 7-62, 7-64
  Command file syntax, 7-59, 18-124
Subset
  Choice of variables, 5-31
Subtotal, 10-81
  Command file syntax, 5-31, 18-133, 18-143
  Meaning of, 10-80
  Printing, 10-82
Subtotals
  Meaning, 10-81
Subtotals results
  Applications, 11-87
  GEMPIE, 8-72, 10-82, 11-87
  Meaning, 11-85
  RunGEM, 11-87
  RunGTAP, 11-87
  SLTOHT, 11-87
  ViewSOL, 8-72, 10-82, 11-87
Suffixes
  For updated data, 4-17
SUP statements in Command files, 7-69
Swapping variables, 5-29
Syntax errors in Command file
  Eliminating, 2-10

T

TABLO Code
  Low memory, 6-57
TABLO style
  Element names, 4-21
TABLO-generated program
  Actions, 6-43
  Equations file, 12-94
  Options Menu, 14-109
TABLO-generated program or GEMSIM
  Choice of variables, 5-31
TABLO-generated programs
  Command file syntax, 18-123
  Fortran 90 or 77, 13-103
  Increasing parameters, 13-105
TABLO-generated Programs
  Command file example, 18-139
TABLO-like statements, 6-56, 6-57
  Qualifiers, 6-57
TABLO-like statments. *See* Command files. *See*
  Command files

Target shocks, 5-37
Tariff rate quotas, 16-120
Terminal data
  Automatic accuracy, 7-66
Terminal data files, 4-17
Terminal writes, 6-46, 14-111
Test Coefficient
  Range, 6-54
TEXTBI, 8-73
Time
  CPU, 2-8, 14-111
Transfer statements, 6-44, 6-56
TRQ, 16-120
TRQmate, 16-120
tshock statements, 5-37

U

U parameter, 12-89
Uniform shocks, 5-35
UPDATE
  Actions, 6-43
Updated data
  Saving after each multi-step calculation, 7-69
Updated data file
  Parameter, 4-17
Updated data files, 4-17
User-specified accuracy. *See also* Automatic accuracy
User-specified Accuracy, 7-64
Using Equations, BCV Files
  Not recommended, 9-77

V

Variable
  Components, 2-9
Variables
  Shocked, 5-31
Variation in results between passes, 12-97
Verbal description, 8-71
  Command file, 18-131

W

W77 option
  TABLO, 13-103
Warning
  Variation in results between passes, 12-97
Windows programs
  RunDynam and RunMONASH, 5-37
Work files
  Cleaning up, 12-99
  May be very large, 12-99
Write
  All steps, 14-111
  Command file, 6-56
Writes, 6-44, 6-45

X

XAC-retained variables, 7-62, 7-65
XDISPLAY, 6-56
XFILE, 6-56
XSET, 6-56
XSUBSET, 6-56
XTRANSFER, 6-56
Xtransfer statements, 6-56
XWRITE, 6-56

Z

Zero coefficients, 12-95, 14-112