Release 6.0 of GEMPACK :

New Features and Changes from Release 5.1 and 5.2

GEMPACK Document No. GPD-4

W. J. Harrison

K. R. Pearson

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

Second edition
October 1998

© Copyright 1985-1998. The Impact Project and KPSOFT.

Second edition
October 1998

ISSN 1030-2514

ISBN 0-7326-1505-4

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 3168, Australia.

Abstract

GEMPACK is a suite of general-purpose economic modelling software especially suitable for applied general equilibrium models. It can handle a wide range of economic behaviour. It also contains powerful capabilities for solving intertemporal models. GEMPACK provides software for calculating accurate solutions of an economic model, starting from an algebraic representation of the model. The equations can be written as levels equations, linearized equations or a mixture of these two.

This document describes the new features in Release 6.0 (October 1998) of GEMPACK and the changes from Release 5.1 (April 1994) and Release 5.2 (September 1996). The new features include

- improved memory management using a Fortran 90 compiler,
- enhancements to SLTOHT for spreadsheet mapping files and for producing tables of results
- reporting levels variables in simulation results
- specifying an acceptable range of values for coefficients, and developments in automatic accuracy
- various speed-ups in solution
- editing mode in ViewHAR
- ViewSOL a Windows viewer for Solution files
- RunGEM, a Windows program for especially simple simulation processing
- enhancements to Set statements in TABLO, including set unions, intersections and null sets, and not having to specify maximum set sizes
- changes to Command file syntax to make Command files easier to write and change
- the ability to write sets during a simulation and to add set and element labelling using MODHAR

Changes from Release 5.1 to Release 5.2 include

- TABLO-like statements on Command files,
- set and element labelling on Header Array file,
- WinGEM which is a Windows interface to GEMPACK on Windows and Windows 95 / NT PCs,
- several new statements allowed in TABLO Input files, and
- improved software for solving models.

A up-to-date list of all Command file statements is included, as are full details of the syntax for GEMPACK text files.

Document Attributes

Name : Release 6.0 of GEMPACK - New Features and Changes from Release 5.1 and 5.2

Audience : CGE Modellers

Identifier : GPD-4

History :	Date	Author(s)	Comment
	Sept 1996	Jill Harrison and Ken Pearson	First Edition
[Title was GPD-4 Release 5.2 of GEMPACK - New Features and Changes from Release 5.1]			
	October 1998	Jill Harrison and Ken Pearson	Second Edition

Table of Contents

1. INTRODUCTION 1-1

1.1 New Features and Enhancements for Release 6.0 1-1

1.2 Changes in GEMPACK Licences and Versions for Release 6.0 1-7

1.3 Documentation for Release 6.0 1-7

1.4 Getting Started with GEMPACK 1-7

1.4.1 Getting Started with GEMPACK and WinGEM on Windows PCs 1-7

1.4.2 Getting Started with GEMPACK on Other Machines 1-8

1.5 New Features and Enhancements for Release 5.2 1-8

1.6 Acknowledgments 1-10

2. COEFFICIENTS AND LEVELS VARIABLES IN SIMULATIONS 2-11

2.1 Reporting Levels Values when Carrying Out Simulations 2-11

2.1.1 How do you see the pre-simulation and post-simulation levels values? 2-13

2.2 Specifying Acceptable Range of Values for Coefficients Read or Updated 2-14

2.2.1 Motivation 2-14

2.2.2 How to Specify the Allowed Range for Values of a Coefficient 2-15

2.2.3 Range Specification Only Relevant for User-specified Accuracy 2-15

2.2.4 Tests Carried Out 2-16

2.2.5 An Example 2-16

2.2.6 Associated New Statements in Command Files 2-17

3. MEMORY MANAGEMENT WITH A FORTRAN 90 COMPILER 3-19

3.1 Increasing Parameters in Fortran 77 3-19

3.2 Automatic Memory Allocation with Fortran 90 3-19

3.3 Fine Print 3-20

3.3.1 Fortran 77 TABLO-generated programs or Fortran 90 TABLO-generated programs 3-20

3.3.2 Preliminary Pass 3-20

3.3.3 Repetitions if memory allocation is too small 3-21

4. RELEASE 6.0 ENHANCEMENTS TO TABLO INPUT FILES 4-23

4.1 Not Specifying Maximum Size of SETs **4-23**

4.2 Set Unions and Intersections **4-24**

4.3 Writing the elements of one set **4-24**

4.4 Writing the elements of all (or many) sets **4-25**

4.5 Enhancements to Semantics for Intertemporal Models **4-25**

5. SET AND ELEMENT LABELLING ON HEADER ARRAY FILES 5-29

5.1 Effect of Set and Element Information on GEMPACK Programs **5-30**

5.1.1 SLTOHT 5-30

5.1.2 SEEHAR 5-30

5.1.3 MODHAR 5-30

5.1.4 TABLO-generated Programs and GEMSIM 5-31

5.1.5 RWHAR and MKHAR 5-31

5.2 Checking Set and Element Information when Reading Data **5-31**

5.3 Adding Set and Element Information to an Existing Header Array File **5-34**

5.3.1 Method 1 - Extra Write Statements 5-34

5.3.2 Method 2 - Using a special TABLO Input file 5-34

5.3.3 Method 3 - Using Updated files 5-35

5.3.4 Method 4 - Using MODHAR 5-35

5.4 Compatability with Earlier Releases of GEMPACK **5-36**

5.5 Technical Point **5-37**

6. RELEASE 6.0 ENHANCEMENTS FOR SLTOHT FOR SPREADSHEETS 6-39

6.1 Option SSS Short Spread Sheet **6-39**

6.2 Enhancements to spreadsheet files written by SLTOHT **6-39**

6.2.1 Element names rather than component numbers 6-39

6.2.2 Output of Tables Using Option SSE 6-40

6.2.3 Output of Levels Results Using Option SSE 6-41

6.2.4 Three and higher dimensional arrays of results 6-41

6.3 Tables Containing Results for Several Variables (SS or SSS output) **6-42**

6.4 Tables Suitable for Producing Graphs **6-45**

6.5 Option SHK Shock CMF output **6-45**

6.6 Programs ACCUM and DEVIA for Dynamic Forecasting Models **6-46**

7. RELEASE 6.0 COMMAND AND LOG FILE CHANGES 7-49

7.1 Using the Command File Stem for Names of Other Output Files **7-49**

7.1.1 Default: Solution file stem = Command file stem 7-51

7.1.2 Default: Log file stem = Command file stem 7-51

7.1.3 Default: Display file stem = Command file stem 7-51

7.1.4 Using <Cmf> in Command files 7-52

7.2 Equations and BCV Files **7-52**

7.2.1 Default: Don't save an Equations file or BCV file 7-53

7.2.2 Default: Model name is blank if unspecified 7-53

7.3 Specifying Command, Stored-input and LOG Files on the Command Line **7-53**

7.4 Stored-input Files, Input Redirection and LOG files **7-54**

8. MISCELLANEOUS CHANGES FROM RELEASE 5.2 TO RELEASE 6.0 8-55

8.1 Speeding Up Simulations **8-55**

8.2 Transferring Long Names when executing Write statements **8-55**

8.3 Automatic accuracy with models which read data from the terminal **8-56**

8.4 Where an ASSERTION Fails **8-56**

8.5 Scaling equations **8-56**

8.6 Shocking a variable with no components exogenous **8-57**

8.7 Saving BCV Files and Using Them **8-57**

8.7.1 Using Existing Equations and BCV Files Not Supported in Future Releases ? 8-57

8.7.2 TABLO Input file written on the Auxiliary files 8-58

8.8 Checking the Closure and Shocks **8-58**

8.9 Additional Shock (ashock) Statements for Dynamic Models **8-59**

9. WINGEM AND RUNGEM ON WINDOWS PCS 9-61

9.1 WinGEM - the Windows Interface to GEMPACK **9-61**
9.1.1 ViewHAR for Looking at or Modifying Data on a Header Array File 9-62
9.1.2 ViewSOL for Looking at Simulation results on a Solution file 9-62

9.2 Getting Started with GEMPACK and WinGEM **9-62**

9.3 RunGEM for Model Users **9-63**
9.3.1 Preparing a Model for Use with RunGEM 9-64

10. TABLO-LIKE STATEMENTS IN COMMAND FILES 10-65

10.1 TABLO-Like Checks of Extra Statements **10-66**

10.2 Other Points **10-67**

11. RELEASE 5.2 ENHANCEMENTS TO TABLO INPUT FILES 11-69

11.1 Set Complements **11-69**

11.2 Sets Whose Elements Depend on Data **11-70**

11.3 Mappings Between Sets **11-70**
11.3.1 Defining Set Mapping Values 11-72
11.3.2 Using Set Mappings 11-74
11.3.3 Index Expressions 11-74
11.3.4 Other Semantics for Mappings 11-74
11.3.5 Writing the Values of Set Mappings 11-75

11.4 Assertions in TABLO Input Files **11-76**

11.5 Index-Expression Conditions **11-77**
11.5.1 Comparing Indices 11-77
11.5.2 Index Expression Comparison 11-77

11.6 $POS Function to Indicate Position Number in a Set **11-78**

11.7 Writing Updated Values from FORMULA(INITIAL)s **11-78**
11.7.1 Saving Updated Values from All FORMULA(INITIAL)s 11-79

12. SOLVING MODELS 12-81

12.1 New Sparse-Linear-Equation Solving Routines MA48 **12-81**
12.1.1 Which to Use? MA48 or MA28 12-81

12.2 Extrapolation Accuracy Summaries for Data and for Levels Variables **12-81**

12.2.1 For Data 12-81
12.2.2 For Levels Variables 12-81

12.3 User-specified Accuracy for Simulations 12-83
12.3.1 Specifying Which Solutions to Use 12-84
12.3.2 Incompatibilities 12-84
12.3.3 Adaptive Stepsize Method 12-84
12.3.4 Several Subintervals (Revisited) 12-84

12.4 Newton's Method for Levels Models 12-85
12.4.1 Newton's method and ORANIF 12-86
12.4.2 Convergence 12-86
12.4.3 General advice 12-87

12.5 Options for Saving Solution and Updated Data Files 12-87

13. OTHER ENHANCEMENTS FOR RELEASE 5.2 13-89

13.1 LOG Files and Early Errors 13-89

13.2 Default Output File Names 13-89

13.3 SEEHAR - New Options 13-89

13.4 GEMSIM and TABLO-generated Programs 13-90
13.4.1 Updated Versions of Parameter Files No Longer Produced 13-90
13.4.2 Displays of Integer Coefficients 13-90
13.4.3 Closure and Shocks in GEMSIM or TABLO-generated Programs 13-90
13.4.4 Solution Method and Step Numbers Added to Verbal Description 13-90
13.4.5 Check that Auxiliary Files are Correct 13-91
13.4.6 Reuse of Pivots in Multi-step Simulations 13-91

13.5 SEENV Program to See the Closure on an Environment File 13-91
13.5.1 Command file output 13-91
13.5.2 Spreadsheet output 13-92

13.6 Closure Specified Using Data Files 13-93

13.7 RWHAR - New Options 13-94

13.8 SUMEQ and the Homogeneity of Models 13-95
13.8.1 Homogeneity Example with Miniature ORANI 13-95

13.9 Displays and Labelled Output from SEEHAR into Spreadsheets 13-98

14. CHANGES FROM RELEASE 5.1 TO RELEASE 5.2 14-99

14.1 Clarification Concerning Shocks on a Command File **14-99**

14.1.1 Shock Statement when only Some Components are Exogenous 14-99

14.1.2 No "shock" statements on a Command file 14-99

14.2 Increasing Parameters in TABLO-generated Programs and GEMSIM **14-100**

14.2.1 Parameters MMNZ and MMNZ1 14-100

14.2.2 Other Parameters 14-100

14.3 Licence Files **14-100**

14.4 Clarification Relating to Closures and Solution Files **14-101**

14.5 Parts of Release 5.1 No Longer Supported **14-101**

14.5.1 TABLO Low Memory Code Options SCS,LIR,LRP 14-101

14.5.2 Submatrix Data Files and Set-up Files 14-101

14.6 Technical Notes for Programmers **14-101**

14.6.1 GPSTRT 14-101

14.6.2 LSTHDS 14-102

15. SYSTEMATIC SENSITIVITY ANALYSIS AND QUOTAS 15-103

15.1 Systematic Sensitivity Analysis **15-103**

15.2 Explicit Volume Quotas and Other Inequalities **15-103**

16. TRANSLATION BETWEEN GEMPACK AND GAMS DATA FILES 16-105

16.1 Transferring Data from GEMPACK to GAMS **16-105**

16.1.1 SEEHAR Method 16-105

16.1.2 TABLO Method 16-105

16.2 Transferring Data from GAMS to GEMPACK **16-107**

17. NEW COMMAND FILE STATEMENTS 17-109

17.1 New Command File Statements for Release 6.0 **17-109**

17.2 New Command File Statements for Release 5.2 **17-110**

18. NEW TABLO SYNTAX, QUALIFIERS 18-111

18.1 New TABLO Statement Qualifiers for Release 6.0 **18-111**

18.2 New TABLO Statement Qualifiers for Release 5.2 **18-111**

18.3 New TABLO Syntax for Release 6.0 **18-111**

18.4 New TABLO Syntax for Release 5.2 **18-111**

19. SUMMARY OF COMMAND FILE STATEMENTS 19-113

19.1 Command Files for GEMSIM and TABLO-generated Programs **19-113**

19.1.1 Command File Statements 19-114

19.1.2 General Points 19-127

19.1.3 Complete Example 19-129

19.2 Command Files for Running SAGEM **19-130**

19.2.1 Complete Example for SAGEM 19-135

20. SYNTAX OF GEMPACK TEXT DATA FILES 20-137

20.1 The "How Much Data" Information **20-137**

20.1.1 Comments 20-139

20.2 Data Values in Text Files **20-140**

20.2.1 Array Sizes 20-140

20.2.2 Actual Data Values of the Array 20-140

20.2.3 Real Numbers 20-141

20.2.4 Character Strings 20-142

20.2.5 Multi-dimensional Real Arrays 20-142

20.2.6 Maximum Length of Lines of a File 20-142

20.2.7 Differences Between Row Order and Spreadsheet Style Data 20-142

20.2.8 Element Name/Number Labels 20-144

20.3 Converting Release 4.2.02 Files for Use with Release 5 or Later **20-144**

21. CHOOSING SETS OF VARIABLES INTERACTIVELY 21-147

21.1 **Specifying Components of One Variable** **21-148**

21.2 **The Lists Option - Choosing a Set Actively** **21-148**

21.3 The Some Option - Responding to Prompts **21-151**

22. REFERENCES 22-153

23. GEMPACK DOCUMENTS **23-155**

CHAPTER 1

1. Introduction

1.1 New Features and Enhancements for Release 6.0

The new features and enhancements of Release 6.0 (October 1998) over and above Release 5.2 (September 1996) relate mainly to the following areas.

(1) **Reporting levels results** at the same time as linearized ones

It is now 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. To do this, you may need to add appropriate information in your TABLO Input file to tell TABLO what are the pre-simulation levels values of selected linear variables. For example, in the linearized TABLO Input file SJLN.TAB for the Stylized Johansen model (see section 3.5.1 of GPD-1), add the qualifier "(ORIG_LEVEL=DVCOM)" when declaring the variable p_XCOM, as in

```
Variable (ORIG_LEVEL=DVCOM) (All,i,SECT)  p_XCOM(i) ;
```

[Here DVCOM(i) is a dollar value for each commodity i. But its original (ie pre-simulation) value can be thought of as a quantity, where one unit of volume is whatever amount can be purchased for one dollar at pre-simulation prices.]

It is also possible to give a numerical value as the original levels value. For example, in SJLN.TAB it would be possible to add

```
Variable (ORIG_LEVEL=1)  (All,i,SECT)  p_PCOM(i) ;
```

to indicate that the original levels value of the commodity prices are being taken as 1.

Then, when you run a simulation, GEMSIM or the TABLO-generated program will report the pre-simulation, post-simulation levels values of XCOM and PCOM, also the change in them, as well as the percentage change p_XCOM in XCOM and p_PCOM in PCOM, for each commodity. See section 2.1 for details.

(2) **Specifying a range for coefficient values**

It is possible to declare a **specified range** within which each coefficient must stay (for example if X(i) > 0 this is written as a qualifier "GT 0" for a coefficient that should always be positive). Checks that the coefficient stays within this range are carried out at run time when you are using automatic accuracy. [If it goes outside the range during a given subinterval, the subinterval is repeated with a shorter length.] See section 2.2 for details.

(3) Improved **memory management** using features available under Fortran 90

When GEMPACK programs are used in conjunction with a suitable Fortran 90 (as distinct from Fortran 77) compiler, you no longer need to increase program parameters which are too small for your model. The GEMPACK programs now use the memory management features available in Fortran 90 to allocate just the amount of memory needed to complete each task. See chapter 3 for details.

[Users with a Fortran 77 compiler and a source-code version of GEMPACK are still catered for with this release; however, the programs still contain parameters which may need to be increased.]

(4) **Not specifying MAXIMUM SIZE** when declaring SETs

When SETs are declared in a TABLO Input file, specifying MAXIMUM SIZE is now not necessary in several cases.

(a) If you have a Fortran 90 compiler, neither TABLO-generated programs nor GEMSIM need MAXIMUM SIZE for sets read in at run time.
(b) If you have a Fortran 77 compiler and are using GEMSIM, MAXIMUM SIZE is not needed for run-time sets (but it is required for TABLO-generated programs).

Details are in section 4.1 below.

(5) **Extended syntax with intertemporal models** can be used which allows indices such as "t+1" to be used in sensible circumstances in which they were not allowed in Release 5.2. See section 4.5 for details.

(6) **Set unions and intersections** are now available

Unions and intersections of sets are now allowed in TABLO Input files. For example, if set A and set B are already defined, the union and intersection of these sets can be defined using the syntax:

```
SET C = A union B
```

```
SET D = A intersect B
```

See section 4.2 for details.

(7) Various **speed-ups** have been made in the way simulations using automatic accuracy or using the MA28 routines are carried. See section 8.1.

(8) **Transferring long names** when Write statements are executed.

When writing an array to a Header Array file, if the long name to use is not specified in the TABLO Input file, TABLO-generated programs and GEMSIM transfer the long name (if not all blanks) from when this same data was read. The intention here is to make it easier for modellers to **preserve long names** on files when doing data organisation tasks. See section 8.2.

(9) **Writing the elements of sets**

The elements of any set can now be written to either a text file or a Header Array file. For example, the TABLO statements

```
Write (SET) COM to file hafile header "COM " ;
Write (SET) COM to file textfile ;
```

would write the elements of the set COM to a Header Array file (the one with logical name "hafile") and to a text file (the one with logical name "textfile") respectively (assuming the relevant logical files had previously been declared to be new files of the appropriate type).
See section 4.3.

You can now **write the elements of all sets** to a Header Array file via a statement of the form

```
Write (ALLSETS) to file hafile ;
```

This is especially useful if you want to add set and element information to a Header Array file via MODHAR (see section 4.4 below).

(10) **Adding set and slement information** using MODHAR

Set and element information can now be added to Header Array files using MODHAR and the 'at' command. In order to do this there is a new option in MODHAR 'ds' Define Sets which allows you to define the element names for a set. There is also an addition to GEMPACK Text file syntax to add the coefficient name usually associated with the array and the sets over which coefficient ranges. For example, the "how much data" information can be

```
26  2  25 real header "BAS1" longname "Intermediate Input data"
coefficient V1BAS(COM, SOURCE, IND) ;
```

See section 5.3.4 below.

(11) **Checking set, element and coefficient 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 5.1.2), these programs can now check whether this set/element/coefficient information as held on the file agrees with what they expect.

This checking will pick up errors that would occur if, for example, the order of the elements in the TABLO Input file was different from that intended when the data was assembled onto the Header Array file.

Command file statements make it easy for you to specify whether or not you want this checking to be done.

See section 5.2 for details.

(12) **New options and enhancements for SLTOHT**

(a) The new option SSS allow you to make a **"short" spreadsheet** or CSV file containing no comments, blank lines or "how much data" information. See section 6.1. The other new option SHK converts values read from a Solution file into "shock" statements for use in CMF files, (section 6.5)

(b) The default is now for SLTOHT to use **element names** rather than component numbers in output spreadsheet files. Component numbers (the only possibility in earlier releases) can be obtained via option NEL. See section 6.2.1 for details.

(c) Option SSE can now produce spreadsheet tables with row and column labels when only part of a variable is specified on the Spreadsheet Mapping file. See section 6.2.2 below.

(d) Whenever option SS or SSS is used to produce a spreadsheet style output file, it is now easy to produce **tables of results** in which results for different variables are in the columns. See section 6.3 for details.

(e) The tables produced with SSE or SS or SSS options should be ideal for making graphs of results (including with intertemporal models). See section 6.4.

(f) GEMPACK-style comments (beginning with a single exclamation mark !) are now allowed in both Header Array and Spreadsheet Mapping files. Blank lines are ignored in both of these types of files.

(13) **Empty sets** are now allowed.

The purpose of this is to let a single TABLO Input file cover more possibilities. For example, in ORANI there is a set of exogenous investment industries and its complement (the other industries); now either this set or is complement could be empty (so either all or none of the industries may be exogenous investment industries). In GTAP the set of endowment commodities is split up into the sluggish and mobile ones; at present, there must be at least one of each, but with the new release, either set can be empty. [We are grateful to Kevin Hanslow for pointing out the usefulness of this facility.]

(14) **Command file streamlining**

Command files list various file names such as Solution or Log file names which by convention are often the same as or similar to the name of the Command file. To facilitate this convention, and avoid unnecessary editing of files, various default file names have been introduced.
For example, if you omit the **Solution file = ...;** statement from the Command file say **sjlb.cmf**, the Solution file will be called the same name as the Command file **sjlb.sl4** by default. See section 7.1 for details.

(15) **Specifying a Command file or Stored-input file on the command line (under DOS)**

The DOS command

```
sj  -cmf  sjlb.cmf
```

will run the TABLO-generated program SJ.EXE and direct it to take its inputs from the Command file SJLB.CMF. This is an alternative (which will be especially useful in DOS batch files) to

responding to the prompts and then indicating “cmf” and then “sjlb.cmf”. You can also use the commands

```
sj   -cmf  sjlb.cmf    -log   sjlb.log
gempie   -sti  gempie.sti    -lon  gempie.log
```

The first takes input from Command file SJLB.CMF and sends output to LOG file SJLB.LOG (as well as to the terminal) whereas the second runs GEMPIE taking inputs from the Stored-input file GEMPIE.STI and sends output just to the LOG file GEMPIE.LOG (not also to the terminal).

See section 7.3 for details about these command line options **-cmf**, **-sti**, **-log** and **-lon**.

(16) **Programs RWSOL and MKSOL**

A pair of utility programs call RWSOL and MKSOL have been written to transfer Solution files from one machine to another, similar to the pair of programs RWHAR and MKHAR used to transfer Header array files. The idea was that Solution files on Unix machines could be transferred to the PC to be viewed with the Windows program VIEWSOL. See section 8.4.3 of GPD-1 for details.

(17) **GEMPACK licence files**

SAGEM now requires a GEMPACK licence. When running with a large model, TABLO-generated programs require a GEMPACK licence; see section 1.2.5 of GPD-1.
See also section 1.2 below.

(18) **ViewSOL**

ViewSOL, a Windows program for viewing Solutions has been developed by Mark Horridge. See section 9.1.2 for details. ViewSOL can also be used to view Levels results if they are present on the Solution file.

(19) **Editing mode in ViewHAR**

ViewHAR, the Windows viewer for Header Array files, has been extended to include an editing mode. Details are given in the Online help for ViewHAR.

(20) **Windows program RunGEM for simulations**

RunGEM is a windows program developed to make it easy to carry out simulations with any model implemented and solved using GEMPACK. It makes it easy for users with no previous experience of GE modelling to solve models. It is also aimed at providing an environment which will increase the productivity of experienced users who need to carry out many different simulations with the same model.

It is easy to prepare a model for use with RunGEM.

Details about RunGEM can be found in chapter 9.

(21) **TABmate**

TABmate written by Mark Horridge is designed to be used for modifying and debugging TABLO Input files. It is aimed at the intermediate or advanced GEMPACK user and is particularly helpful for large or complex TAB files.

(22) **Programs ACCUM and DEVIA**

Programs ACCUM and DEVIA (based on Mark Horridge's ACCUMSOL and DEVIATE) have been made available as part of GEMPACK for use with dynamic models (such as MONASH and MEGABARE) which are solved several times in a year-on-year fashion to produce a base case forecast or a policy deviation from a base case. ACCUM assembles the results of the different years of these runs into a spreadsheet file while DEVIA produces a spreadsheet file which shows the differences between the policy (deviation) run and the base case.

These programs are described in more detail in section 6.6.

(23) **Additional shocks for policy runs with a dynamic model**

Statements of the form

ashock … ;

(for example, "ashock xtax = uniform 5 ;") can be used to specify additional policy shocks with dynamic models (such as MONASH). See section 8.9 for details.

(24) **Checking the closure and shocks**

It is now possible to ask GEMSIM or a TABLO-generated program to check the closure and shocks statements in any Command file. This will happen quickly, even with a large model. See section 8.8 for details.

(25) **Absorption during condensation**

We are **no** longer supporting **absorption** during condensation (see section 2.3.3 of GPD-2) since we think omitting variables is more useful and we have checked that none of our users is doing absorption.

There is no more detail about points 13 and 25 above. A few other changes from Release 5.2 are listed in chapter 8.

1.2 Changes in GEMPACK Licences and Versions for Release 6.0

Release 6.0 programs will not run with a Release 5.2 or earlier GEMPACK licence.

Details of the current versions of GEMPACK and associated licences can be found in section 1.2 of GPD-1.

SAGEM now requires a GEMPACK licence.

TABLO-generated programs produced with Release 6.0 of GEMPACK now require a GEMPACK licence when they are used to solve large models - see section 1.2.5 of GPD-1 for details.

There is now an Unlimited Executable-image Version of GEMPACK which is available in special circumstances - see section 1.2.3 of GPD-1.

The size limits on the (Limited) Executable-image version of GEMPACK are shown in full detail in section 6.2 of GPD-7. These are also the limits on a TABLO-generated program being run without any kind of GEMPACK licence. If a simulation with a TABLO-generated program exceeds any of these limits, a Large-simulations GEMPACK licence (see section 1.2.5 of GPD-1) is required.

1.3 Documentation for Release 6.0

All of this document should be read in conjunction with the other GEMPACK documentation, namely GEMPACK documents GPD-1, GPD-2, GPD-3 and GPD-8. There is a new edition (the fourth edition) of GPD-1 for Release 6.0. GPD-2 and GPD-3 have not changed from the Release 5.1 editions since the features documented there have not changed except in one or two isolated cases to which we draw your attention at the appropriate places in this document. GPD-8 (see section 1.4 below) is new for Release 6.0.

1.4 Getting Started with GEMPACK

Document GPD-8 contains a hands-on introduction for new users. It also contains details about the models and associated files usually distributed with GEMPACK. The hands-on parts of this document are in two parts, the first for users on Windows PCs (who will want to use WinGEM, the Windows interface to GEMPACK), and the second for users on other machines (for example Unix machines) who work at the command line.[1]

1.4.1 Getting Started with GEMPACK and WinGEM on Windows PCs

If you work on a Windows PC, and are new to GEMPACK, or if you are experienced with GEMPACK but have previously worked at the DOS prompt, you will want to see how WinGEM (the Windows interface to GEMPACK works). The examples in section 2 of GEMPACK Document GPD-8 provide a step-by-step introduction to GEMPACK via WinGEM.

[1] Section 1 of GPD-8 is an updated version of what was Appendix B of GPD-1 in the Release 5.2 documentation. Section 2 of GPD-8 is an updated version of what was Appendix B of GPD-4 in the Release 5.2 documentation. Section 3 of GPD-8 is an updated version of what was Appendix E of GPD-1 in the Release 5.2 documentation.

1.4.2 Getting Started with GEMPACK on Other Machines

If you are using GEMPACK on a machine other than a Windows PC (for example, a Unix machine), you can work through the examples in section 3 of GEMPACK Document GPD-8. These examples provide a step-by-step introduction to GEMPACK.

1.5 New Features and Enhancements for Release 5.2

This section documents the features of Release 5.2 (September 1996) of GEMPACK which were not available in Release 5.1 (April 1994) of GEMPACK.

The following features were new for Release 5.2.

1. When running GEMSIM or TABLO-generated programs, **extra TABLO-like statements** (for example, declaring extra sets or asking for extra WRITEs or DISPLAYs) can be included in the Command file. Such statements are carried out as if they were appended to the end of the original TABLO Input file. [See chapter 10 below for details.]
2. Arrays on Header Array files can now contain **set and element labelling information**, namely the names of the coefficients usually associated with the arrays, the names of the sets over which the arguments of the array run and the names of the elements of the sets involved with the array. TABLO-generated programs and GEMSIM now automatically write this information to any arrays they write to a Header Array file. This information is shown in SEEHAR output and also when SLTOHT produces Header Array output (the default) or when the new option 'SSE' is selected. It is also shown by ViewHAR (see below). [See chapter 5 below for details.]
3. When GEMPACK runs on DOS machines under Microsoft Windows or Windows95, it can be run under a new **GEMPACK Windows interface** called **WinGEM**. Users can carry out modelling tasks using GEMPACK windows. We hope that this will make modelling easier for both new and experienced modellers using these machines. As part of this, a new Windows program ViewHAR (written by our colleague Mark Horridge) allows windows-type access to examine the data in Header Array files. [You can think of ViewHAR as a Windows version of SEEHAR.] [See chapter 9 below for details.]
4. **SET complement** syntax, for example

   ```
   SET NONMARCOM = COM - MARCOM ;
   ```

   is allowed in TABLO Input files. [See section 11.1 below for details.]
5. **Sets** can be defined in TABLO Input files in ways **which depend on data**, as in for example,

   ```
   SET SPCOM = (all,c,COM: TOTX(c) > TOTY(c)/5 ) ;
   ```

   which says that the set SPCOM consists of all commodities c in COM for which TOTX(c) is greater than one-fifth of TOTY(c). [See section 11.2 below for details.]
6. Explicit **mappings between sets** (for example, a mapping from unique-product industries to unique-product commodities, mapping each industry to the commodity it makes) are available in TABLO Input files. [See section 11.3 below for details.]
7. A new **function $POS** is allowed which can be used to indicate the position number of an index or element in a set in TABLO Input files. [See section 11.6 below for details.] .
8. A new type of statement called **ASSERTION** is allowed in TABLO Input files. These allow modellers to request that conditions they expect to hold be checked. For example, the following ASSERTION statement requests the software to check that no entries in a certain array are

negative.

```
ASSERTION # Check no negative values #
                    (all,c,COM) DVHOUS(c) >= 0 ;
```

[See section 11.4 below for details.]

9. New **conditionals involving indices and/or set mappings** explicitly, for example

```
 IF( i EQ j, <expression> ),
```

are available in TABLO Input files. [See section 11.5 below for details.]

10. It is now possible to **write updated** (that is, post-simulation) **values of coefficients initialised via a FORMULA(INITIAL)**. [See section 11.7 below for details.]

11. New **sparse linear equation solving routines MA48** (sometimes a faster alternative to the MA28 routines) are now the default. [See section 12.1 below for details.]

12. You **can now specify in advance what accuracy you require** for simulations carried out via TABLO-generated programs or GEMSIM. You specify what accuracy you require (how many significant figures) and what percentage of results you require to be this accurate and the program works out how many steps and how many subintervals are required to achieve this accuracy. [See section 12.3 for details.]

13. **Newton's method** for solving levels equations is now available.
TABLO is used to add the appropriate Newton correction term to levels equations. When running a simulation using GEMSIM or the TABLO-generated program, the shock to the Newton variable can be added to the usual shocks in an Euler solution, or the Newton shock can be applied alone in a series of Newton correction steps applied after the original Euler multi-step simulation. [See section 12.4 below for details.]

14. 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 recently. Details about these procedures are given in chapter 15 below.

15. A facility for **translating GEMPACK Header Array files to GAMS** format. [There is also a complementary facility available in GAMS for translating GAMS data files to GEMPACK format.] [See chapter 16 below for details.]

16. Various other enhancements are described in chapter 13 below.

1.6 Acknowledgments

We are grateful to Philip Adams, Betina Dimaranan, Soren Frandsen, Kevin Hanslow, Tom Hertel, Mark Horridge, Lars-Bo Jacobsen, Patrick Jomini, Robert McDougall, Ronald Wendner and others (including others of our colleagues at the Centre of Policy Studies and Impact Project) who have beta tested the Release 6.0 software and given us helpful feedback.

We are grateful to Channing Arndt, Christian Bach, Betina Dimaranan, Soren Frandsen, Tom Hertel, Mark Horridge, Michael Malakellis, Robert McDougall, Maureen Rimmer, Padma Swaminathan, Tom Wahl, Peter Wilcoxen and others (including our colleagues at the Centre of Policy Studies and Impact Project) who have beta tested the Release 5.2 software and given us helpful feedback.

CHAPTER 2

2. Coefficients and Levels Variables in Simulations

The features described in this chapter are new for Release 6.0.

This chapter describes how to use GEMSIM or a TABLO-generated program to calculate and report pre- and post- simulation values of levels variables and include them on your Solution file so that they can be printed out on your GEMPIE printout or can be looked at using ViewSOL.

Coefficients and levels variables may naturally fall in a certain range of acceptable values, for example, a coefficient may always be positive. The second section of this chapter describes how to specify an acceptable range of values for coefficients or levels variables so that GEMSIM or a TABLO-generated program can carry out tests to see if this coefficient stays in the specified range (for example, after updating). If the coefficient or variable fails this test, the program can reduce the subinterval length to ensure that the coefficient or variable stays within the specified range.

2.1 Reporting Levels Values when Carrying Out Simulations

It is now 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.[2] To do this, you may need to add appropriate information in your TABLO Input file to tell TABLO what are the pre-simulation levels values of selected linear variables.

For example, in the linearized TABLO Input file SJLN.TAB for the Stylized Johansen model (see section 3.5.1 of GPD-1), add the qualifier "(ORIG_LEVEL=DVCOM)" when declaring the variable p_XCOM, as in

```
Variable (ORIG_LEVEL=DVCOM)  (All,i,SECT)  p_XCOM(i) ;
```

[Here DVCOM(i) is a dollar value for each commodity i. But its original (that is, pre-simulation) value can be thought of as a quantity, where one unit of volume is whatever amount can be purchased for one dollar at pre-simulation prices.]

Also add the qualifier (ORIG_LEVEL=1) when declaring the variable p_PCOM, as in

```
Variable  (ORIG_LEVEL=1)  (All,i,SECT)  p_PCOM(i) ;
```

to indicate that the original levels value of the commodity prices are being taken as 1.

Then, when you run a simulation, GEMSIM or the TABLO-generated program will report the pre-simulation, post-simulation levels values of XCOM and PCOM, also the change in them, as well as the percentage change p_XCOM in XCOM and p_PCOM in PCOM, for each commodity.

[2] This is true even if your TABLO Input file has only linear variables and linearized equations.

The syntax for these variable qualifiers are

```
(ORIG_LEVEL=<coefficient-name>)
```

or

```
(ORIG_LEVEL=<real-number>)
```

Qualifiers "ORIG_LEVEL=" are only needed when you define linear variables (percentage changes or changes).

In a levels or mixed model, you do not need such qualifiers when declaring levels variables (since the correspondence is clear). [When you declare a levels variable X, TABLO does not need to be told that the pre-simulation values of the associated linear variable p_X or c_X are given by the coefficient X.] Thus, for example, no change is needed to the mixed TABLO Input file SJ.TAB for Stylized Johansen (see section 3.3.2 of GPD-1) in order to get levels values reported when you carry out a simulation.

Note that no indices need to be shown (nor will TABLO allow them to be shown) in a "ORIG_LEVEL=" qualifier. Simply specify the name of the coefficient. TABLO automatically checks that the coefficient and variable have the same numbers of indices and that these indices range over exactly the same sets; a semantic error will be reported otherwise.

The coefficient referred to in a qualifier

```
(ORIG_LEVEL=<coefficient-name>)
```

must be a real coefficient (not an integer one).

To make it easier to add "ORIG_LEVEL=" qualifiers to existing TABLO Input files, we have made an exception to the usual rule that everything must be defined before it is referenced. In the qualifier

```
(ORIG_LEVEL=<coefficient-name>)
```

the Coefficient in <coefficient-name> is allowed to be declared later in the TABLO Input file. For example, in SJLN.TAB (see section 3.5.1 of GPD-1), it is correct to add the qualifier "(ORIG_LEVEL=DVHOUS)" to the declaration of variable p_XCOM even though coefficient DVHOUS is not declared until several lines later.

Note that the values of <coefficient-name> must, of course, be available (via a Read or Formula). The values needed are just the pre-simulation values - the values are not needed (for this purpose, at least) at subsequent steps. Thus, if you are adding Formulas just to give the values in question (and not also for other purposes), you can make them Formula(Initial)s. For example, in SJLN.TAB it is natural to add the qualifier (ORIG_LEVEL=Y) to the declaration of linear variable p_Y. Coefficient Y (value of household expenditure) was not available in the Release 5.2 version of SJLN.TAB. In the Release 6.0 version we have added it via

```
Coefficient  Y  # Total nominal household expenditure # ;
Formula (Initial)  Y = SUM(i, SECT, DVHOUS(i) ) ;
```

[Had we not done so, we could not add the (ORIG_LEVEL=Y) qualifier when declaring variable p_Y.]

2.1.1 How do you see the pre-simulation and post-simulation levels values?

When you run GEMPIE and select the totals solution, you will see the levels values reported unless you select GEMPIE option "**NLV**" (No levels values)[3]. Then, for each component of each variable whose pre-simulation levels values are know (e.g., via an ORIG_VALUE= qualifier), you will see 4 numbers underneath each other, as in, for example,

3.000 (the percent change)

500.0 (the pre-simulation levels value)

515.0 (the post-simulation levels value)

15.0 (the change from pre-simulation to post-simulation)

The linearized result (percentage-change or change) is always first, then below it are the pre-simulation, post-simulation and change results.

Note that levels values are not necessarily available for all variables. [In a linearized TABLO Input file, they are only available for those variables for which you add an "ORIG_LEVEL=" qualifier when you declare the variable.]

If you don't want to include these levels results on a Solution file, you can add the statement

```
levels results = no ;
```

to your Command file.

The current versions of ViewSOL also show any levels results. SLTOHT processes levels results if present if you specify the SLTOHT option **SHL** - Show levels results, if available. The default is not to show levels results so that the new SLTOHT is still compatible with old Stored Input files or batch programs written for Release 5.2.

[3] Levels results are not shown if you select GEMPIE option **SNA** (Single solution not across the page) since levels results are only shown when the results go across the page.

2.2 Specifying Acceptable Range of Values for Coefficients Read or Updated

2.2.1 Motivation

It is now possible to tell the software that certain coefficients should never be negative. The software is then able to produce more accurate results for multi-step simulations because it is able to ensure that these specified coefficients do indeed never become negative throughout the simulation.

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 2.2.1). 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 2.2.1, 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 12.3), 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.

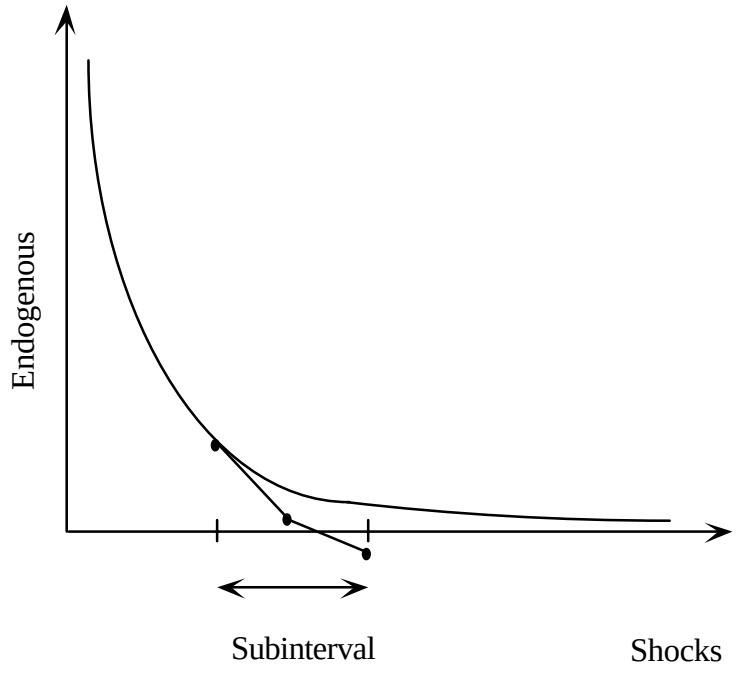

Figure 2.2.1 : Levels value can become negative if subinterval is too long

2.2.2 How to Specify the Allowed Range for Values of a Coefficient

In your TABLO Input file, new Coefficient qualifiers which specify acceptable range of values for a coefficient are now allowed. For example,

```
Coefficient (GE 0) (all,c,COM) DVHOUS(c) ;
```

specifies that DVHOUS(c) values should always be at least equal to zero.

These qualifiers are also allowed in the declaration of a LEVELS Variable as in, for example,

```
Variable (Levels, GE 0) (all,c,COM) DVHOUS(c) ;
```

In this case, 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 >
LE 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 only be attached to Coefficients which are READ and/or UPDATEd (though the software allows them to be attached to any Coefficients).

They can also be attached usefully to any Levels Variable (since these are always updated).

For example,

```
Variable (Levels, GE 0, LT 1.5) (all,i,IND) LABIND(i) ;
```

2.2.3 Range Specification Only Relevant for User-specified Accuracy

Range specifications only affect simulation results if you use user-specified accuracy (see section 12.3). [Readers not familiar with this idea are advised to at least skim that section before proceeding here.] That is, these qualifiers are not taken into account unless your Command file contains the statement

```
automatic accuracy = yes ;
```

This is because, otherwise (if you specify the number of steps and subintervals), the software is not able to adjust the subinterval length, and so cannot back-track when something goes outside the specified range.

2.2.4 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

If your simulation is using user-specified accuracy (that is, your Command file includes the statement "automatic accuracy = yes ;"), these range qualifiers are used to test the UPDATEd value of the coefficient in question (at each step and whenever its values are extrapolated).

This test of updated values is only carried out if you are using user-specified accuracy.

If an updated value falls outside the specified range, the current subinterval is redone with a shorter length.

2.2.5 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.[4]

If this same simulation were carried out with a "GE 0" qualifier for VXMD and automatic accuracy were used, the software would recognise this value and automatically adjust the step size to avoid having any negative values during the calculations and extrapolations.

---

[4] 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 new "GE 0" qualifier would not be 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.

2.2.6 Associated New Statements in Command Files

These are only relevant if "automatic accuracy = yes ;" is in the Command file. The first two below are relevant even if there are no range restrictions in the TABLO Input file. The last two below are only relevant when there are range restrictions attached to one or more Coefficient or Levels Variables in the TABLO Input file.

(1) **`minimum subinterval length = * ;`** ! default is 0.000001 (10^{-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.

(2) **`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.

(3) **`range test updated values = updated|extrapolated|BOTH|no ;`**
! 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.

(4) **`range test initial values = YES|no ;`** ! 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.

CHAPTER 3

3. Memory Management with a Fortran 90 Compiler

The features described in this chapter are new for Release 6.0.

The features of Fortran 90 allow greatly improved memory management compared to Fortran 77. In order to take advantage of these capabilities, the main GEMPACK programs such as SAGEM, SEEHAR, and also GEMSIM and TABLO have been rewritten using dynamic array allocation available in Fortran 90.

There are now 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 disks but the appropriate programs installed on your computer differ depending on whether you have a Fortran 77 or Fortran 90 compiler.

The modelling functionality of 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.

3.1 Increasing Parameters in Fortran 77

If you run GEMPACK programs written for Fortran 77, you may receive messages saying the value of one of the program parameters must be increased (see section 5.5 of GPD-1). For example,

```
**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)
```

These messages occur when some aspect of your model has become too large for the program you are running. This is particularly annoying in long batch jobs which then failed to complete in time for your deadline.

3.2 Automatic Memory Allocation with Fortran 90

Users with a suitable Fortran 90 compiler (for example, the Lahey LF90 compiler on PCs) will find that they no longer get these 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.][5]

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 can fit more often within the memory available. When GEMPACK is installed on a network, the same GEMPACK

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

program can be run on machines of small and large memory sizes, and larger machines can still run large problems without the need to adjust program parameters.

Apart from the MAXIMUM SIZE change described in section 4.1, 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.

3.3 Fine Print

Below are a couple of small points for users interested in a little more detail. [If any of these points seems obscure, please just ignore it.]

3.3.1 Fortran 77 TABLO-generated programs or Fortran 90 TABLO-generated programs

Any version of TABLO can write output for GEMSIM or can write a TABLO-generated program (as before).

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 while
- the "Fortran 90" program does its own memory management.

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.

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 new 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.

3.3.2 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 11.2).]

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).

3.3.3 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 get the memory correct to carry out LU decompositions. 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 3.3.3..
```

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.

CHAPTER 4

4. Release 6.0 Enhancements to TABLO Input Files

This chapter describes the enhancements made to TABLO Input files in Release 6.0. The first section spells out a useful consequence about set sizes for those using a Fortran 90 compiler. The second section is about specifying set unions or intersections. The next two sections describe how to write the elements of one or all sets defined in a TABLO Input file to a Header Array file or text file; these may be useful in adding set and element to a Header Array file using MODHAR (as described in section 5.3.4 below). The final section describes more flexible syntax and semantics for intertemporal models.

Note that the new SET and WRITE statements in sections 4.2, 4.3 and 4.4 below can also be used as "extra" XSET and XWRITE statements in Command files (see chapter 10).

4.1 Not Specifying Maximum Size of SETs

In many cases, in TABLO Input files, it may not be necessary to specify the MAXIMUM SIZE of SETs.

For example, in GTAP the statement

```
SET REG # Regions # MAXIMUM SIZE 10
  READ ELEMENTS FROM FILE GTAPSETS Header "H1  " ;
```

can be replaced by

```
SET REG # Regions #
  READ ELEMENTS FROM FILE GTAPSETS Header "H1  " ;
```

This means it is no longer necessary to have different TABLO Input files for different sized versions of a model. [Another example is with the Monash Model; now the 25-sector aggregation used in the Monash Model Course can have the same TABLO Input file as the 113-sector used by CoPS.]

More precisely, the cases where you can omit the MAXIMUM SIZE are

(i) if you have a Fortran 90 compiler and are writing a TABLO-generated program, or

(ii) if you have any compiler and are writing output for GEMSIM.

In these cases, any MAXIMUM SIZEs in the TABLO Input file are ignored by TABLO. [For example, you may say in the TABLO Input file that a set IND has maximum size 10 and then read in, quite happily, a set of size 20 when you run GEMSIM or the Fortran 90 TABLO-generated program.]

However, if you have a Fortran 77 compiler and are writing a TABLO-generated program, you still need to specify MAXIMUM SIZEs of all sets. TABLO checks this when you ask to write a TABLO-generated program in this case.

Because of this, there is a new option **RMS** [Require Maximum Set Sizes] for the CHECK stage of TABLO. You should select this option if you have a Fortran 77 compiler and plan to write a TABLO-generated program at the CODE stage. When this option is selected the statement

```
SET REG # Regions #
  READ ELEMENTS FROM FILE GTAPSETS Header "H1  " ;
```

(see above) would produce a semantic error. If RMS is not selected there would still be an error later in the TABLO run, because then TABLO would not find the error until the CODE stage.

4.2 Set Unions and Intersections

Unions and intersections of sets are now allowed in TABLO Input files. For example, in GTAP, once the sets ENDW_COMM (endowment commodities) and TRAD_COMM (tradeable commodities) are defined, the set DEMD_COMM (demanded commodities) can be defined via the statement

```
Set DEMD_COMM # Demanded Commodities # = ENDW_COMM union TRAD_COMM ;
```

The syntax for set union and intersection is

```
SET <set-name> [#<labelling-information>#] = <set-name1> UNION <set-name2> ;
SET <set-name> [#<labelling-information>#] = <set-name1> INTERSECT <set-name2> ;
```

Here both <set-name1> and <set-name2> must be previously defined sets whose elements are known at run-time (either by being specified in the TABLO Input file, or read from a file at run-time, or inferred from other sets as, for example, a set defined as the complement or union of two sets with known elements).

The elements of the **UNION** are those in <set-name1> followed by those in <set-name2> which are not in <set-name2>.

The elements of the **INTERSECTion** are those in <set-name1> which are also in <set-name2>.

For example, if

SET1 has elements (e1, e2, e3, e4) and
SET2 has elements (e2, e6, e1), and if

```
SET3 = SET1 UNION SET2 ;
SET4 = SET1 INTERSECT SET2 ;
SET5 = SET2 UNION SET1 ;
```

then

SET3 has elements (e1, e2, e3, e4, e6),
SET4 has elements (e1, e2), and
SET5 has elements (e2, e6, e1, e3, e4).

Note that, although the elements of SET3 and SET5 are the same, their order is different. The order is, of course, important in associating data with indices.

Intertemporal sets are not allowed in Set Union or Intersection statements.

We are grateful to Robert McDougall for suggesting that TABLO allow explicit unions and intersections.

4.3 Writing the elements of one set

The syntax of the new TABLO statement to do this is

```
WRITE (SET) <setname> to FILE <logical-file> [HEADER "<header>"] ;
```

Here the HEADER part is required if <logical-file> has been previously declared to be a new Header Array file. The HEADER part is not allowed if <logical-file> has been previously declared to be a new text file. In the former case (Header Array file case), the output is written as strings of length 12 (the length of set element names). In the second case (Text file case) the output is written using the standard GEMPACK syntax for character strings (see chapter 20).

When writing to a Header Array file, the long name written is of the form

Set <setname> <labelling information>

This ensures that the file produced will be of maximum use with the 'ds' option in MODHAR (see section 5.3.4 below).

Note that this syntax (without a HEADER part) was allowed in Release 5.2 if <logical-file> had been declared to be a new GAMS file (see chapter 10 of GPD-4). This functionality remains unaltered.

We are grateful to Mark Horridge for suggesting this feature.

4.4 Writing the elements of all (or many) sets

The syntax is

```
WRITE (ALLSETS) to file <hafile> ;
```

If you are running a Fortran 90 version of GEMSIM or a TABLO-generated program, this will write the elements all sets whose elements are specified. If you are running a Fortran 77 version of these programs, it will only write out the elements of those sets which have been defined earlier in the TABLO Input file (whose elements have been specified).

The file in question must be (the logical name of) a Header Array file. No other writes are allowed to this same file (in order for us to be able to ensure relatively easily that the resulting file will not have duplicate headers).

If the elements of a set have been read from a Header Array file, they will be written to the same header (where possible); otherwise the program makes up a header. Do not include a header name in the WRITE(ALLSETS) statement.

The long name written is of the form

```
Set <setname> <labelling information>
```

This ensures that the file produced will be of maximum use with the '**ds**' option in MOHDAR (see section 5.3.4 below).

Note that this statement can be added to your Command file as an "extra" statement. For example, the following statements in a Command file will write out all (or most) sets to the file somesets.har.

```
xfile (new) manysets ;
file manysets = somesets.har ;
xwrite (allsets) to file manysets ;
```

4.5 Enhancements to Semantics for Intertemporal Models

Up till (and including) Release 5.2 (September 1996), there have been some limitations in the semantics of expressions allowed in Formulas, Equations and Updates.

Example 1

```
Set (Intertemporal) S1_6 Size 6 ( p[1] - p[6] ) ;
Set (Intertemporal) S2_4 Size 3 ( p[2] - p[4] ) ;
Coefficient (All,s,S1_6) C1(s) ;
Coefficient (All,s,S1_6) C2(s) ;
Read C2 from Terminal ;
Formula (All,s,S2_4) C1(s) = C2(s+1) ;
```

In this example, the last statement (the Formula) would produce a semantic error indicating that 's' on the RHS is expected to range over a subset of the set S1_6. In the example above this could be easily fixed by adding the statement

```
Subset S2_4 is subset of S1_6 ;
```

Example 2

```
Set (Intertemporal) S1_6 Size 6 ( p[1] - p[6] ) ;
Set (Intertemporal) S2_10 Size 9 ( p[2] - p[10] ) ;
Coefficient (All,s,S1_6) C1(s) ;
Coefficient (All,s,S2_10) C2(s) ;
Read C2 from Terminal ;
Formula (All,s,S1_6) C1(s) = C2(s+1) ;
```

With Release 5.2 this would generate a similar semantic error, namely that index 's' is expected to range over a subset of the set S0_6 (which it does not).

But "clearly" the last Formula in example 2 makes perfect sense (except to Release 5.2 TABLO) since 's' runs from p[1] to p[6] and so "s+1" runs from p[2] to p[7] all of which are in the domain S2_10 of coefficient C2.

In this second example, there is no way to "fix" the Formula by adding a Subset statement. Indeed S1_6 is not a subset of S2_10. [What is really happening is that adding "+1" to the elements of S1_6 always gives an element of S2_10.]

The enhancements are to allow Formulas such as the last one in Example 2.

The enhancements may also allow you to reduce the number of Subset statements. The purpose of Subset statements it to facilitate index checking (see section 4.5.2 of GPD-2). When indices in Formulas, Equations and Updates range over Intertemporal sets which have intertemporally-defined elements, associated Subset statements (to facilitate index checking as set out in section 4.5.2 of GPD-2) are now not required.

TABLO tries to work out if the arguments (with or without an index offset) are valid. If it is unable to, it gives instructions to the TABLO-generated program or to GEMSIM to check at run-time that indices stay in range (taking into account offsets, if they are present). [See section 7.2.1 of GPD-2 for information about intertemporal sets which have their elements defined intertemporally (as distinct from those intertemporal sets which have fixed elements).]

Of course the two sets in question must have the same intertemporal element stem (see section 7.2.1 of GPD-2): for example, the formula in Example 1 would generate a semantic error if the set S2_4 were defined by

```
Set (Intertemporal) S2_4 Size 3 ( t[2] - t[4] ) ;
```

since its elements have stem 't' and those in S1_6 have stem 'p'.

However Subset statements are still required if the intertemporal sets in question have fixed elements. Thus, if the Set declarations in Example 1 were changed to

```
Set (Intertemporal) S1_6 Size 6 ( p1 - p6 ) ;
Set (Intertemporal) S2_4 Size 3 ( p2 - p4 ) ;
```

the formula in Example 1 would not be allowed (unless S2_4 was declared as a subset of S1_6). If the Set declarations in Example 2 were changed to

```
Set (Intertemporal) S1_6 Size 6 ( p1 - p6 ) ;
Set (Intertemporal) S2_10 Size 9 ( p2 - p10 ) ;
```

the formula in Example 2 would not be allowed and this could not be "fixed" by a subset statement since set S1_6 is not a subset of S2_10.

Subset statements are still required in conjunction with Reads, Writes and Displays, even when the sets involved are Intertemporal sets which have intertemporally-defined elements. Thus, the following statement

```
Write (all,s,S2_4) C2(s) to terminal ;
```

added at the end of Example 1 above will still produce a semantic error unless C2_4 is declared to be a Subset of C0_6. [Maybe future enhancements to TABLO will automatically generate the required Subset statement in cases such as this.]

Because Subsets are required for Reads and because Formula(Initial)s generate a Read statement (for steps 2,3... of a multi-step calculation), Subsets are still required for Formula(Initial)s.

Even when the sets in question have intertemporal elements, and subset statements are not required to facilitate Equations etc, subset statements may still be a good idea if you want to refer to different sets on Command files. For example, if you are working with Example 1 above modified so that the Coefficients there are Linear Variables (and the Formula there is an Equation) then you could say

```
exogenous C1(S2_4) ;
```

would only be allowed if S2_4 had been declared as a subset of S1_6 in the TABLO Input file.

CHAPTER 5

5. Set and Element Labelling on Header Array Files

Arrays on Header Array files can contain set and element labelling information, namely the names of the coefficients usually associated with the arrays, the names of the sets over which the arguments of the array range and the names of the elements of the sets involved with the array. TABLO-generated programs and GEMSIM automatically write this information to any arrays they write to a Header Array file. This information is shown in SEEHAR output and also in various forms of SLTOHT output. It is also shown by Mark Horridge's ViewHAR program (see chapter 9).

This set and labelling information was first allowed with Release 5.2 of GEMPACK. The material in sections 5.2 and 5.3.4 is new for Release 6.0. The rest of the material in this chapter has not changed since Release 5.2.

Below is an example of labelled output from SEEHAR showing the element names for the rows and columns.

Example of labelled output from SEEHAR

```
Next submatrix contains numbers from the ATILDE(SEC,SEC) part, which
  are positions (1-8, 1-8) of ATILDE(SEC,SEC).
  COLUMN                1           2           3           4
  ROW

                  agriculture    mining    food_process   light_ind
   1 agriculture    82.779999    84.540001    46.580002    25.650000
   2 mining         30.709999    25.969999    11.670000     1.100000
   3 food_process   10.040000    10.830000    97.760002    16.059999
   4 light_ind      25.450001     3.820000    13.320000    33.320000
   5 heavy_ind       1.150000     0.490000     1.420000     1.070000
   6 machinery       0.700000     0.490000     0.890000     0.560000
   7 social          3.560000     5.000000     7.030000    11.110000
   8 services       11.940000    10.840000    30.990000    14.820000
   COLUMN
   TOTALS          166.329987   141.979996   209.660019   103.689995
```

We refer to this labelling information as **set and element information** on an array.

This chapter describes how to put this information onto an existing (or new) Header Array file and explains various related changes to GEMPACK programs. However old Header Array files without this set and element information can still be used in the usual way.

5.1 Effect of Set and Element Information on GEMPACK Programs

Arrays which contain set and element labelling are said to be of type **RE**
(whereas arrays without this labelling are of type **RL** or **2R**).

The type of each array is shown when SEEHAR or ViewHAR list the contents of a Header Array file. Note that set and element labelling can only be attached to real data at present, not to integer (or character) data.

5.1.1 SLTOHT

See chapter 6 to see how SLTOHT uses this information. In particular, it is simple to prepare tables of results or data with element labels using SLTOHT option SSE (see section 6.2.2).

5.1.2 SEEHAR

When SEEHAR runs, it uses the set and element names in its output (except if you choose option SS for spreadsheet output). Its default (labelled) output for arrays with this information is now almost identical to that from a DISPLAY statement in a TABLO Input file (see the example above). In the SEEHAR summary of Header Arrays, arrays of type **RE** contain set and element information whereas real arrays without this information are of type **RL** or **2R**.

If you want set and element information on spreadsheet output, you can choose the option SSE. If you import SSE output into a spreadsheet, it will produce a table with labels on the rows and columns. Careful choice of elements names can be used to produce a table suitable for reports. If you need to change the original element names to something suitable for reporting, Method 2 in 5.3 below can be used to put new element names for the sets, still retaining the original coefficient values from your data file.

Example of SEEHAR output for option 'SSE'

```
! Real array of size 3x4x3 from header "EVFA"
!
! Coefficient from which this array was written
! 'EVFA(ENDW_COMM:PROD_COMM:REG)'.
!
! (This array is shown as 3 matrices - each of size 3x4.)
! The matrix EVFA(%1:%2:"USA")
!  with %1 in ENDW_COMM and %2 in PROD_COMM. ++++++++++++++++++++++
              ,food         ,mnfcs        ,svces         ,CGDS       ,
  Land        ,   17125.5   ,  0.000000   ,  0.000000    ,  0.000000 ,
  Labor       ,   96441.2   ,   792128.   ,  2.606860E+06,  0.000000 ,
  Capital     ,   97376.8   ,   382360.   ,  1.408053E+06,  0.000000 ,
```

5.1.3 MODHAR

When MODHAR is used to modify the data on an array containing set and element information, the resulting modified array retains this information except when the size of the array is changed (for example, when the size is increased using the 'is' subcommand in MODHAR, or part of an array is written using the 'wp' subcommand) when no set and element information is put on the modified array.

It is possible to add arrays with set and element information using MODHAR's option 'ah', to a file containing arrays with or without this information. The arrays are each self-contained units.

With Release 6.0, data on a text file used to add arrays to a Header Array file via MODHAR's 'at' command can have set and element labelling information automatically attached - see section 5.3.4 for details.

5.1.4 TABLO-generated Programs and GEMSIM

When TABLO-generated programs or GEMSIM write arrays to a Header Array file (via a WRITE statement), they write the set and element information. In simulations which produce updated data files which are Header Array files, TABLO-generated programs and GEMSIM write the set and element information even if the original data file from which the data was read did not contain the set and element names.

With Release 6.0, when these programs read arrays which contain set and element information, they are able to check that the coefficient, set or element names agree with those on the TABLO Input file - see section 5.2 for details.

5.1.5 RWHAR and MKHAR

The programs RWHAR and MKHAR preserve any set and element information.

5.2 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), these programs can now 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 or:

- 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.

Now GEMSIM and TABLO-generated programs already check that the amount of data at the header is as expected. 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). But Release 5.2 software was not able to check the four things above.

For example, if 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[6]) but (since the second sector on the file is services), presumably this second number really refers to services and not manufactures.

Release 6.0 software can now check whether all of the four things above (coefficient name, number of arguments, names of sets, names of elements) expected 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.

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.]

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

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 = warn ;
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 wish to return to the Release 5.2 situation where no checking is done, 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.[7] 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.

[7] 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.

5.3 Adding Set and Element Information to an Existing Header Array File

Methods 1 to 3 below were available in Release 5.2. Method 4 below (which uses MODHAR and relies on the TABLO enhancements in sections 4.3 and 4.4 above) is new for Release 6.0.

5.3.1 Method 1 - Extra Write Statements

At present the easiest way of adding the set and element information to an existing Header Array file, is to add extra write statements to a Command file which already solves the model (or at least reads in the data). For example the following statements could be put at the front of SJLB.CMF (see Figure 2.3.1 in GPD-1) to produce a new version SJNEW.DAT of the base data file SJ.DAT for the Stylized Johansen model.

```
!
! Add the following to SJLB.CMF to write SJ.DAT with set
!  and element information.
!
neq = yes ; ! No equations and so don't do a simulation
!
! Set up the new version
!
xfile (new) sjnew # New HA file containing set element info # ;
file sjnew = sjnew.dat ;
!
! Write the data to the new file (SJ.TAB already contains
!     instructions to read the data)
!
xwrite DVCOMIN to FILE sjnew HEADER "CINP" ;
xwrite DVFACIN to FILE sjnew HEADER "FINP" ;
xwrite DVHOUS  to FILE sjnew HEADER "HCON" ;
!
! End of additions to Command file
!
```

With these statements added to SJLB.CMF, you can run GEMSIM or the TABLO-generated program for Stylized Johansen (without having to rerun TABLO) and produce a new version of the Header Array file which has set and element labelling information on it. [To check this, you could run SEEHAR on the new data.] Of course the actual data on the file is the same as on the original file; but set and element labelling is added to it.

With Release 6.0, note that you do not need to include the LONGNAME in the XWRITE statements since long names are now automatically transferred (see section 8.2).

5.3.2 Method 2 - Using a special TABLO Input file

Another way of adding the set and element information to an existing file is to write a special-purpose TABLO Input file which declares the sets and coefficients, reads the coefficient values from an old file not containing set and element information and then writes the coefficients to a new file.

5.3.3 Method 3 - Using Updated files

Another way of adding the set and element information to an existing Header Array file is to run a simulation in which all the shocks are zero. Hence the data in the updated data files is the same as in the original data files but the updated data files now contain the set and element information.

5.3.4 Method 4 - Using MODHAR

The method described here (which uses MODHAR and relies on the TABLO enhancements in sections 4.3 and 4.4 above) is new for Release 6.0.

It is possible to add set and element information using MODHAR when adding arrays from a text file using the '**at**' option.

To do this there are four steps:

1. **Prepare** a Header Array file containing **the element names** of the relevant sets as character Header Arrays. Details of the format for character string data is given in chapter 20. Alternatively you can use a WRITE(ALLSETS) or XWRITE(ALLSETS) statement (see section 4.4) to prepare the element name data.
2. When preparing the text file containing the GEMPACK text arrays, **add the coefficient name and the sets or elements over which the coefficient ranges** to the "how much data" information at the top of each array (see chapter 20)**.** This is only applicable to real arrays. The "how much data" information can now have the form

   ```
   <sizes> <type> <order> <header> <longname> <coefficient> ;
   ```

   <coefficient> is optional and has the format

   ```
   coefficient <coefficient-name>(<set-name>,<set-name>,....)
   ```
   or
   ```
   coefficient <coefficient-name>(<element-name:set-name>,...)
   ```

   Examples of this format are given below.
3. **Run MODHAR** and select the new menu option '**ds**' **Define Sets**. You will be asked for the name of the Header Array file containing the element names for the sets. Then you will be asked for a list containing the set name and the Header of the array which lists the element names for that set. (A carriage return ends the list.)
   For example, suppose the elements of set COM are in the array with header "CCCC" and the elements of set IND are in header "IIII" the list would be

   ```
   COM  CCCC
   IND  IIII
   <carriage-return>
   ```

   If on the Header array file containing the element names, you have arrays with longname starting with the word "Set" and with the following word the name of the set that you are defining eg COM, for example, if the longnames of two of the Header Arrays are

   "Set COM Commodities"
   "Set IND Industries"

   then MODHAR will select these Headers and associate them with the sets COM and IND respectively. You will be asked if you wish to add all sets with longnames of this type as a group

without your having the enter the names of the sets and associated Headers one by one. This type of longname "Set COM ...." is the form of output produced in a TABLO-generated program or GEMSIM by the statement (see section 4.4)

```
Write (ALLSETS) to file hafile ;
```

You need to define the sets prior to referring to them in step 3.

4. Select the **'at' option in MODHAR** and add the array from the text file in the usual way.

Examples of arrays on text file

- The coefficient V1LAB is defined over the set of industries IND.

```
5  real  row-order  header "LAB1"  longname "Labour data"
coefficient  V1LAB(IND) ;
1.1  2.0  4.4  5.5  0.3
```

- The array at header "BAS1" contains domestic intermediate input data for the coefficient V1BAS defined over the sets COM, SOURCE and IND.

```
3  1  2 real  header  "BAS1"  longname "Intermediate Input data"
coefficient  V1BAS(COM, "dom" : SOURCE, IND) ;
1.1  2.2
4.5  5.6
5.0  6.0
```

MODHAR will check that the sets COM, SOURCE and IND have been defined previously using the 'ds' command. If so, set and element information will be written when the associated data is written to the Header Array file being created (that is, the array will have type 'RE'); if not, MODHAR will write out the data without any set and element information (that is, an 'RL' array will be written).

5.4 Compatability with Earlier Releases of GEMPACK

All Release 5.2 and later GEMPACK programs can read all Header Array files produced using Release 5.1 (or earlier) software.

But Release 5.1 GEMPACK programs cannot read Header Array files containing set and element labelling information produced by Release 5.2 programs. (If they attempt it, they will almost certainly crash with an error.)

To enable Release 5.2 and 6.0 users to send their Header Array files to other GEMPACK users who only have Release 5.1, we have provided an option with the program RWHAR which enables you to produce a Header Array file with the same data but with the set and element labelling information removed.

To copy a Header Array file with set and element labelling information to a Header Array file with no set and element labelling information, run RWHAR (the Release 5.2 or 6.0 version) and select option

C51 Convert to Release 5.1 Header Arrays (no set/element labels)

When prompted by RWHAR, input the name of the original file (with set and element information) and then the name of the new Header Array file (which will not contain this information)..

We will make this Release 6.0 version of RWHAR freely available on our Web and FTP sites.

5.5 Technical Point

The set and element information is stored separately with each array on a Header Array file. This means that programs can deal with arrays one at a time and can "cut and paste" between files as usual (for example, by using option 'ah' in MODHAR as was described above).

There is no requirement that the sets be consistent for the different arrays on a file. [For example, there is no consistency check to prevent one array relating to a set COM with 23 elements and another array on the same file relating to a set COM with 45 elements.]

CHAPTER 6

6. Release 6.0 Enhancements for SLTOHT for Spreadsheets

This chapter describes two new options for SLTOHT, namely SSS (see section 6.1) and SHK (see section 6.5), and extensive improvements in the use of Spreadsheet Mapping files [see section 8.3 of GPD-1]. These changes make components of variables easier to choose .

There are also several enhancements to the spreadsheet files produced by SLTOHT.[8]

SLTOHT can also be used to produce tables of variables in which results for different variables are in the columns.

Dynamic models (such as MONASH and MEGABARE) are solved several times in a year-on-year fashion to produce a base case or policy deviation from a the base case. Results from these different runs are assembled into spreadsheets using programs ACCUM and DEVIA which are described in section 6.6 below. This material is included in this chapter about SLTOHT since ACCUM and DEVIA operate on spreadsheet files produced by SLTOHT.

6.1 Option SSS Short Spread Sheet

The idea behind the option **SSS** is to avoid having to produce spreadsheet output such as a CSV file and then to need to run a separate program such as SHORTCSV.EXE to remove comments, blank lines and other unwanted material before you can put the results in a spread sheet.

Option SSS produces text file output similar to option SS. However no comment lines, no "how much data" information or blank lines are included. You can choose the separator between data values as a comma or some other character.

You must use a spreadsheet mapping file to choose the variables in your output file.

6.2 Enhancements to spreadsheet files written by SLTOHT

6.2.1 Element names rather than component numbers

This section applies when options SS or SSS are selected.

In the past, when SLTOHT has written out the simulation results for one or more variables in spreadsheet form, component numbers have followed the variable name, as in, for example,

```
y(1), 23.4 ,
y(2), -3.2,
x(1), 10.8,
```

Now, by default, element names are used (when they are available) instead of component numbers, as in, for example,

[8] See also section 13.9 for a way of making tables in a spreadsheet program using SEEHAR.

```
y(food), 23.4,
y(services), -3.2,
x(food), 10.8,
```

If the variable in question has two or more arguments these will appear, as in, for example,

```
y(food,dom), 23.4,
y(services,dom), -3.2,
```

If you prefer component numbers, you can still see these if you select option "NEL" from SLTOHT's options menu.

6.2.2 Output of Tables Using Option SSE

Option SSE produces spreadsheet tables with row and column labels. It is especially aimed at variables with two or more indices.

An example comes from the GTAP model with

3 tradeable commodities food,mnfcs,svces, (the set TRAD_COMM)
3 regions USA,EU,ROW, (the set REG)
3 endowments Land,Labor,Capital, and
CGDS which is capital goods,
and the set NSAV_COMM has elements (Land, Labor, Capital, food, mnfcs, svces, CGDS)

The variable **qo(i,r)**, with i in NSAV_COMM and r in REG, is the percent change in the output of each commodity i in each region r.

In a spreadsheet mapping file, the line

qo

would lead to the following table (once the file produced by SLTOHT is imported into a spreadsheet program).

Table of qo(NSAV_COMM,REG)

	USA	EU	ROW
Land	0.00	0.00	0.00
Labor	0.00	0.00	0.00
Capital	0.00	0.00	0.00
food	0.89	-0.35	-0.11
mnfcs	-0.15	0.09	0.03
svces	0.00	0.81	0.00
CGDS	-0.00	-0.00	-0.01

[This feature was part of Release 5.2 of GEMPACK.]

Now, **with Release 6.0, you can specify subsets (and elements)** on the Spreadsheet Mapping file if you are only interested in a (rectangular) part of a variable. For example, the line

qo(TRAD_COMM, REG)

in a Spreadsheet Mapping file would lead to a similar table but with only the rows for the commodities (food,mnfcs,svces) in the **subset TRAD_COMM of the set NSAV_COMM** over which the first index of qo ranges.

Table of qo(TRAD_COMM,REG)

food	0.89	-0.35	-0.11
mnfcs	-0.15	0.09	0.03
svces	0.00	0.81	0.00

For the 3-dimensional variable **qxs(i,r,s)** [exports of commodity i from region r to region s], the line

qxs(TRAD_COMM,"EU",REG)

in a Spreadsheet Mapping file will produce a 2-dimensional table showing exports of all commodities in TRAD_COMM from Europe "EU" to all other regions.

By way of comparison, the line

qxs

by itself would produce 3 such tables, firstly the table of all exports to the USA, then to EU and finally to the rest of the world "ROW". [Three and higher dimensional arrays are broken up under option SSE into the relevant matrices, with the 3rd and higher index changing from matrix to matrix.]

6.2.3 Output of Levels Results Using Option SSE

If levels results are available (see section 2.1 above), SLTOHT considers them to form a new set which it calls "**Lin+Lev**" with the elements "Linear", "PreLev", "PostLev" and "Changes". You will see this with the Stylized Johansen model if you put the line

p_XCOM

into a Spreadsheet Mapping file and use this to process a Solution file containing levels results for the 1-dimensional variable p_XCOM(i) where i ranges over the set SECT=(s1,s2) of commodities. The results produced by SLTOHT would look something like

Percentage and Levels Results for p_XCOM

	Linear	PreLev	PostLev	Changes
s1	5.885	8.000	8.470	0.470
s2	6.899	12.000	12.828	0.828

The "Linear" results are the percentage changes, then come the Pre-simulation and Post-simulation levels, and then the changes in the levels values from pre- to post-simulation.

When levels results are available, a 2-dimensional variable produces a 3-dimensional array, and so on.

6.2.4 Three and higher dimensional arrays of results

Consider again the example of GTAP variable **qxs(i,r,s)** reporting export volume percent changes where i is the commodity (ranging over the set TRAD_COMM of tradeable commodities), r is the origin and s the destination (where r and s range over the set REG of regions). If you put "qxs" on a line of a Spreadsheet Mapping file, the output will be several TRAD_COMMxREG tables (one for each destination region).

Suppose that you preferred several tables showing source and destination, one for each commodity. You could achieve this via the following line in a Spreadsheet Mapping file.

qxs(trad_comm,%reg,%reg)

Here the % characters indicate the row and column index set for each of the tables (matrices) output; you can have at most two such in any one line.

If you are reporting levels results, you may prefer to have all four lots of results (linear and the 3 levels results) together in each table. For example, the line

qo(trad_comm,%reg)

in a Spreadsheet Mapping file will produce the qo output results in several tables whose rows are indexed by the set REG (the first % set) and whose columns are indexed by the different results (Linear, PreLev, PostLev and Changes). [It is as if you have put "qo(trad_comm,%res,%Lin+Lev)" in your Mapping file, though this would be technically illegal.]

In general, you can use these % characters before a set name to indicate how you want 3 and higher dimensional arrays of results to be grouped in tables.

6.3 Tables Containing Results for Several Variables (SS or SSS output)

It is now possible to specify on a **Spreadsheet Mapping file** (see section 8.3 of GPD-1) that you want results for several variables to appear side by side, as in, for example,

Table of Several Variables

```
     , u    ,    y ,    EV
USA,  0.31 ,  0.26, 2345096
EU ,  0.02 , -0.01,    2345
```

where results for the three variables u, y, EV [all with one index which ranges over a set containing regions USA,EU (etc)] are shown side by side in a table. Rows are labelled by components and columns by variable names.

To achieve the above, simply put the line

u : y : EV

in your Spreadsheet Mapping file.

These tables are only available if you have selected option '**SS**' or '**SSS**' from SLTOHT's menu options (not if you select option 'SSE').

The general syntax for the spreadsheet mapping file, for producing these tables is of the form

VAR1 [arguments] : VAR2 : VAR3 (etc)

where VAR1, VAR2, VAR3... are the names of variables and the arguments of interest can be specified optionally.

The simplest case is where no arguments are specified explicitly and all variables have arguments ranging over exactly the same sets (as in the first example above where each of the variables u,y,EV has just one argument ranging over the set REG of regions). This is still allowed even if each variable on the line has two or more arguments; in that case, you will see output of the form

```
                ,  qst  ,  qds
(food:USA)  ,  3.4  ,  2.6
(svces:USA) ,  0.1  ,  0.2
```

(etc) where the element names from the two sets are separated by colons '**:**' (since the spreadsheet would interprete commas as starting a new cell). [Here qst and qds each has 2 arguments ranging over the sets TRAD_COMM and REG.]

If you only want some of the results for the variables in question, you can specify arguments to say which subset you are interested in. For example, suppose that variables **x4** and **sales** both have a single argument ranging over the set COM of all commodities and that MARGCOM is the subset of margins commodities. Then the line

```
x4 (MARGCOM) : sales
```

on a Spreadsheet Mapping file will produce results for just the commodities in MARGCOM.

Another time when you may want to specify the arguments of interest explicitly is if the variables in question range over different sets, where a subset is involved. Suppose, for example, that variable **x4** has one argument ranging over COM and that variable **marsales** has one argument ranging over MARGCOM. Then the line

```
x4 (MARGCOM) : marsales
```

on the Spreadsheet Mapping file will produce results for just the commodities in MARGCOM for the two variables x4 and marsales.

A more realistic example comes from the GTAP model with 3 tradeable commodities food,mnfcs,svces and 3 regions USA,EU,ROW, when the line

```
qo (trad_comm,reg) : qst
```

in a Spreadsheet Mapping file would lead to the following table (once the file output by SLTOHT is imported into a spreadsheet program).

Table of qo(TRAD_COMM,REG) : qst

	qo	qst
(food:USA)	0.89	0.00
(mnfcs:USA)	-0.15	0.00
(svces:USA)	0.00	0.00
(food:EU)	-0.35	0.00
(mnfcs:EU)	0.09	0.00
(svces:EU)	0.81	0.19
(food:ROW)	-0.11	0.00
(mnfcs:ROW)	0.03	0.00
(svces:ROW)	0.00	0.15

In the above table, qo results are percentage changes in output of the relevant commodity in each region and the qst results are percentage changes in sales of these commodities from these regions to the international transport activity. [Note that the arguments "(trad_comm,reg)" are necessary since without them qo(NSAV_COMM,REG) and qst(TRAD_COMM,REG) would not have the same size (since TRAD_COMM is a subset of NSAV_COMM).]

Formal rules

(i) The line

```
VAR1 : VAR2 : VAR3 etc
```

(without explicit arguments specified) is legal provided that

(a) each of the variables has the same number of arguments, and

(b) each argument of the later variables VAR2,VAR3 etc is either equal to, or has as a subset, the corresponding argument of VAR1.

For example, if VAR1 has two arguments ranging over the sets COM1 and REG1 and VAR2 has two arguments ranging over the sets COM2 and REG2 then COM1 must be equal to or a subset of COM2 and REG1 must be equal to or a subset of REG2. In this case the table will just contain the (COM1,REG1) results for each variable.

(ii) The line

VAR1 (arguments) : VAR2 : VAR3

(with explicit arguments specified) is legal provided that

(a) each of the variables has the same number of arguments, and

(b) each argument of every one of the variables VAR1,VAR2,VAR3 etc is either equal to, or has as a subset, the corresponding argument in the arguments specified.

For example,

```
VAR1 (COM1,REG1) : VAR2 : VAR3
```

is legal if each of VAR1,VAR2,VAR3 has two arguments which range over COM1 and REG1 respectively, or else has these sets as a subset. [For example, if VAR2 has two arguments ranging over the sets COM2 and REG2 then COM1 must be equal to or a subset of COM2 and REG1 must be equal to or a subset of REG2.] The table will contain the (COM1,REG1) results for the variables.

It is also possible to use **component numbers** on the next line rather than arguments to specify the components of interest. For example, the two lines

```
u : y : EV
  2  3  ;
```

in a Spreadsheet Mapping file will produce a table with the results for components 2 and 3 of variables u,y,EV.

When component numbers are specified on the next line, arguments must not be specified explicitly on the line containing the variable names; that is, (SSA_REG is a subset of REG)

```
u (SSA_REG) : y : EV          !Wrong
  2  3 ;
```

is not allowed.

Component numbers are allowed on the next line only if each of the variables in question has arguments ranging over exactly the same sets. Thus, for

```
VAR1 : VAR2
  2  3 ;
```

to be allowed, if VAR1 has two arguments ranging over the sets COM1 and REG1, VAR2 must also have two arguments ranging over the same sets (subsets are not allowed).

General principle

Another (more abstract) way of thinking of the above rules is

1. If arguments are not present, they are implied to be those for the sets over which the indices of VAR1 range. Then the components of the variables which are placed in the table are those corresponding to these explicit or implied arguments,
2. except when all variables have exactly the same arguments and the components numbers wanted follow on the next line.

In the former case (1), it should then be clear why each of the arguments of VAR2,VAR3 etc must equal or contain as a subset the explicit or implied arguments of VAR1.

Element Names

In all cases of these tables, element names will be used (if available) to label the rows (even if you select option 'NEL') because component numbers could be ambiguous.

6.4 Tables Suitable for Producing Graphs

Example 1. Suppose you want to graph the regional-indexed variables y,u,EV from GTAP. The table produced with option SS (or SSS) via the line

```
y : u : EV
```

will be easy to convert into a graph showing these three variables for each region.

Example 2. Consider an intertemporal model. This will have several variables indexed just over the time set. The table produced with option SS (or SSS) via a line in a Mapping file which lists the relevant variables separated by colons will produce a table which is easy to convert into a graph showing these variables over time.

Example 3. Consider the variable qo(TRAD_COMM,REG) in GTAP. The table produced with option SSE and the line "qo" in a Mapping file will be easy to convert into a bar chart showing percent changes in output for each commodity in each region (provided you don't have too many of either commodities or regions - if you do, set up subsets say COM1 and REG1 of major interest and put the line "qo(COM1,REG1)" in your Mapping file).

6.5 Option SHK Shock CMF output

This is an option developed for use with the MONASH model (a forecasting model of the Australian economy developed by our colleagues at the Centre of Policy Studies). When carrying out policy simulations (deviations from a base forecast) with MONASH, it is necessary to re-run simulations carried out originally with one closure using a second closure (referred to as the deviation or policy closure). In these re-runs, all exogenous variables in this second closure must be given shocks equal to their values (exogenous or endogenous) from the original simulation. The option SHK prepares the shocks part of the Command file for this re-run with the second closure. [This option SHK may also be useful with other models where simulations are re-run with different closures.]

Option **SHK Shock CMF output** produces the required shock statements directly. Previously to prepare shocks for such a deviation simulation with the Monash model it was necessary to firstly produce CSV output from a Solution file using SLTOHT, and then run a DOS program called SS2SHOCK.EXE to convert the CSV output to statements

```
shock <variable-name> <component> = <value> ;
```

For example,

```
shock x2 3 = 3.1 ;
```

where value 3.1 is the value read from the solution file for component 3 of variable x2.

Option SHK produces text file output without any comment lines apart from one comment giving the name of the original solution file.

The Solution chosen must contain just one column of results, usually the cumulative total solution.

A spreadsheet mapping file must be used to select the variables in your output file. (If you want to output all the variables on the Solution file, you can use the GEMPACK program SEENV with the Solution file and choose "spreadsheet" output to create a map of all variables.)

6.6 Programs ACCUM and DEVIA for Dynamic Forecasting Models

Programs ACCUM and DEVIA are useful with dynamic models (such as MONASH and MEGABARE) which are solved several times in a year-on-year fashion to produce a series of Solution files. [9]

Dynamic models such as MONASH and MEGABARE are solved to produce

- a **base case** forecast for a set of years in sequence (eg 1998, 1999, 2000,…2003)
- a **policy deviation** from this base case in response to some (additional) policy shocks, again for the same sequence of years.

The results of the simulations are several Solution files, one Solution file for each year of the base case, and corresponding Solution files (one for each year) in the policy deviation from the base case. A natural way of presenting results for the base case is a spreadsheet table where the different columns represent different years (eg 1998, 1999,….) and the rows of the table are the results for the variables (in the Solution files). A similar table can be produced for the results of the policy deviation. A final step is to prepare a separate table which reports (for each year) how different the results of the policy deviation are from those of the base case.

ACCUM assembles the results of the different years of these runs into a spreadsheet file while DEVIA produces a spreadsheet file which shows the differences between the policy (deviation) run and the base case.

Using ACCUM to prepare a spreadsheet table for the forecast

The usual method of preparing a spreadsheet table for the forecast is as follows:

1. Run SLTOHT with the each of the Solution files for the base case simulations. Use the same Spreadsheet Mapping file for all of them and use option "**SSS**" (with a comma as separator) to prepare a CSV file for each year included in the forecast. [See sections 6.1 and 6.3 for information about option SSS and Spreadsheet Mapping files.]
2. Run the program ACCUM to assemble the yearly CSV files for the different base case years into a single CSV file for the whole period (eg 1998,1999,2000,…2003). ACCUM will ask you for the number of short spreadsheet files to accumulate and their filenames.
3. Import the accumulated CSV file into a spreadsheet.

A table similar to the one below is produced.

[9] Mark Horridge's programs ACCUMSOL and DEVIATE, which are written in Pascal, only work on PCs. We have rewritten them in fortran and made them part of GEMPACK to extend the use of these utilities to GEMPACK users working on Unix and other operating systems. We are grateful to Mark Horridge for permission to include these useful utilities as part of GEMPACK.

Table of Forecast Results

Variable	run98	run99	run00	run01	run02	run03	Total	Geom total	Ave annual
cr	3.60	4.10	3.20	4.10	3.00	3.00	21.00	22.92	3.50
impvol	8.92	7.97	7.27	8.64	6.13	6.02	44.94	54.19	7.48
expvol	8.00	8.00	10.00	8.00	7.00	6.23	47.23	57.51	7.87
cpi	2.10	3.60	3.60	3.40	3.00	3.00	18.70	20.21	3.12
phi	-1.94	-2.05	-3.31	-1.65	-2.09	-2.10	-13.14	-12.45	-2.19
toft	3.49	-1.54	3.35	-2.93	0.00	0.00	2.37	2.22	0.37
gdpreal	4.27	3.95	3.33	3.82	3.02	2.89	21.27	23.24	3.54
employ	3.30	2.40	2.30	3.40	2.00	2.00	15.40	16.41	2.57
gdp	7.40	7.19	7.94	6.61	6.27	6.14	41.55	49.42	6.92

If you wish, ACCUM calculates the three final columns which contain the arithmetic total of the preceding columns, the geometric total and the average annual growth rate. ACCUM knows which variables are percentage-change variables and which are ordinary change variables. If the variable is known to be a change variable, only arithmetic totals and averages are used. For percentage-change variables, the geometric total is found in the second final column and the final column gives the average geometric growth rate. The actual formula used for the last three columns are given below:

```
For a variable x(j),    where  j  =  1, 2,… N (the number of years)

Arithmetic Sum of x(j) = Sum(j = 1,..N,  x(j))
     This is the column headed “Total”.

Geometric Product = Product(j = 1,.. N,  (1 + x(j)/100))

Geometric Total = 100 [Geometric Product  -  1]
     This is the column headed “Geom Total”

Arithmetic Average  = Arithmetic sum / N
     This is the column “Ave annual” for change variables

Average Annual Growth Rate = 100 [ Nth root of Geometric Product  - 1]
     This is the column “Ave annual” for percentage-change variables
```

A similar table is produced for the deviation results from the separate years of the deviation solutions.

Using DEVIA to prepare a spreadsheet table for the differences

DEVIA produces a spreadsheet file which shows the differences between the policy (deviation) run and the base case. The differences calculated by DEVIA, z(j), where j=1, 2, ….N (the number of years) are given by the formulae below.

```
For percentage values of  x(j), (the base case)       where  j = 1, 2 ….N
                          y(j), (the deviation),

the value z(j) shown in the spreadsheet produced by DEVIA is

    z(j)  = 100  Product {k=1,…,j,  (1 + y(k)/100)/(1 + x(k)/100)} - 1]

for a percentage-change variable.

For an ordinary change variable, the value z(j) shown in the spreadsheet
produced by DEVIA is

    z(j) =  Sum{k = 1,…,j,  (y(k) - x(k)) }
```

The values z(j) calculated by DEVIA are therefore **cumulative differences to the end of the j th year**, caused by the policy shocks.

The input to DEVIA are the filenames of the accumulated base case CSV file and the accumulated policy deviation CSV file. DEVIA assumes that the last three columns in both CSV files are the three Totals columns calculated in ACCUM, so DEVIA ignores these columns.

DEVIA produces a table similar to the one below. [We show just a few rows.]

Table of Cumulative Differences between Policy Deviation and Base Case

Control	**run98**	**run99**	**run00**	**run01**	**run02**	**run03**
Deviation	**pol98**	**pol99**	**pol00**	**pol01**	**pol02**	**pol03**
cr	1.03	2.08	3.14	4.63	4.72	4.80
impvol	0.46	0.98	1.40	1.87	1.93	2.01
expvol	0.73	1.49	2.30	3.05	3.16	3.23
cpi	1.15	2.41	3.52	4.73	4.84	4.99

The top two rows show the names of the CSV files from which the base case (Control) and policy deviation results were obtained in making the table. The values in the rows are the cumulative differences which are calculated by the formulae given above. For example, the figure 2.08 for variable **cr** in the column for year 1999 is the cumulative difference over the two years 1998 and 1999.

CHAPTER 7

7. Release 6.0 Command and Log File Changes

Since Command files were introduced with Release 5.0 of GEMPACK (April, 1993), we have noticed changes in use, and seen experienced users develop conventions which assist them with file management and reconstruction of earlier work. In Release 6.0, we have made some changes to the way the software interacts with Command files which will, we hope, make it easier for all users to follow what seems to be good practice (as explained in the section below).

7.1 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.

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, this meant editing your file to change the Solution file name and the names of other output files for every change in closure or shocks. In Release 6.0, the need for much of this extra routine editing has been eliminated by changes in the way the software works.

This chapter outlines these changes, which apply to Command files used with GEMSIM, TABLO-generated programs or SAGEM. Your old Command files will still run without changes but we hope these new defaults will make your Command files easier to write and change.

As indicated above, in the Command file SJLB.CMF for a simulation with Stylized Johansen, you will often see the lines

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

We have introduced new defaults into Release 6.0 to make it easier to follow the naming conventions described above.

In Release 6.0,

(a) if you don't include a line "`solution file = ... ;`", the Solution file name is taken from that of the Command file.

(b) you can use the statement **`updated file iodata = <cmf>.upd ;`** to indicate that you want the name of the updated file to be based on the name of the Command file. [Here the <cmf> 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][10]

(c) you can include a statement **`log file = yes ;`** to indicate 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".

Thus, in Release 6.0, the above three lines could be replaced by

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

Neither of these needs to be changed when you copy this file as the basis of a new simulation.

Note that the new defaults in points (a), (b) and (c) 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, namely

(d) 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 that affects you.]

The new defaults should not affect existing Command files which should still function as before. With the new defaults you can still specify Solution, LOG or Display file names to be different from that of the Command file if you wish by using the same statements as you would have done under Release 5.2.

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.

---

[10] 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.

This only applies to Command files which end in the suffix **.CMF** or **.cmf** (any case is allowed). We refer to the part of the Command file name without the suffix .CMF as the **Command file stem**.

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.

We set out the new defaults more formally below.

7.1.1 Default: Solution file stem = Command file stem

If the statement **`solution file = <file_name> ;`** is not present, the Solution file name uses the stem of the Command file name, provided the Command file ends with suffix *.cmf* or *.CMF* (any case is allowed). This is applicable for Command files used with GEMSIM, TABLO-generated programs or SAGEM.

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

7.1.2 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*). This is applicable for Command files used with GEMSIM, TABLO-generated programs or SAGEM.

For example if you put the statement *log file = yes ;* in the Command file ***sjlb.cmf*** the log file would be called ***sjlb.log***. The exception to this is if you have a statement *log file = <file_name> ;* (where <file_name> is any file name other than "yes"). In this case your log file would be called "file_name".

7.1.3 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** (This is provided the Command file ends with suffix *.CMF* as before). This is applicable for Command files used with GEMSIM and TABLO-generated programs.

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***.

7.1.4 Using <Cmf> in Command files

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

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

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

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

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

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

7.2 Equations and BCV Files

Creating Equations and BCV (Base Coefficient Values) files was the normal custom when Command files were introduced. These files have now become less important.

An Equations file is still needed if you want to use SUMEQ to help analyse homogeneity (see section 13.8) or a singularity, or if you wish to use SAGEM to carry out Johansen simulations. But it is not necessary to produce one routinely when you do a simulation with GEMSIM or a TABLO-generated program.

A BCV file is only useful if you wish to speed up a subsequent simulation by starting from existing Equations and BCV files (see section 19.1.1). This speedup has become less important and starting from existing Equations and BCV files disables some desirable features (for example, levels results are not available if you start from existing Equations and BCV files, nor are "extra" statements allowed - see section 10.2). [Indeed, we are seriously considering not allowing simulations to start from existing Equations and BCV files in the next release of GEMPACK - see section 8.7.1.]

To reduce the number of output files produced routinely (and to save a little on disk space) we have changed the defaults so that Equations and BCV files are only produced when you include statements asking for them, that is

```
equations file = ... ;
BCV file = ... ;
```

statements. If you leave out either or both of these statements, the corresponding file will not be produced. [This is essentially the same as making "NSE = yes ;" and "NSC = yes ;" the defaults - see section 19.1.1]

Another deprecated feature in GEMPACK is the use of a MODEL name (as in "model = ... ;") to go on Equations files and Environment files (see section 7.7 of GPD-1). 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. We describe these new defaults more formally below.

7.2.1 Default: Don't save an Equations file or BCV file

In Release 6.0, you can omit the `equations file = <file_name>;` statement. In this case, no Equations file or BCV file is saved after the simulation is finished. (In Release 5.2, this was available using the option "NSE = yes ;".)

In Release 6.0, **not saving an Equations file** is the default for GEMSIM or TABLO-generated programs. However you may need the Equations file to use with SUMEQ or SAGEM.

An Equations file is only saved if you include a statement `equations file = <file_name> ;` in your Command file.

Even in this case where there is an `equations file = ... ;` statement, a BCV file is not saved after the simulation. This is different from Release 5.2 when a BCV file was saved whenever there was an '`equations file = ... ;`' statement.

This is the one case where these new Release 6.0 defaults may cause changes in your old Command files: where you want to have both the Equations file and the BCV file saved from a run of GEMSIM or the TABLO-generated program. To save both of these files, you need both the statements

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

in your Command file. Once upon a time, we used to save these files in order to speed up simulation in GEMSIM or TABLO-generated program by reusing the equations file with a statement

```
use equations file = ... ;
```

.

However modern computers are so fast, this method is rarely used. So the current default situation is not to save either an Equations file or a BCV file. (The default now is equivalent to in Release 5.2 putting both `NSE = yes ;` and `NSC = yes ;` in your Command file.)

Note that we may not support the use of Equations and BCV files to start a simulation in the next release of GEMPACK - see section 8.7.1.

7.2.2 Default: Model name is blank if unspecified

You can also omit the `model = <model_name> ;` statement from your GEMSIM/TG-program Command file. In previous Releases of GEMPACK this would have caused a syntax error. Now the program supplies a model name equal to all blanks. If there is a *model = ... ;* statement, the model name functions in the same way as previously.

7.3 Specifying Command, Stored-input and LOG Files on the Command Line

The DOS command

```
sj  -cmf  sjlb.cmf
```

will run the TABLO-generated program SJ.EXE and direct it to take its inputs from the Command file SJLB.CMF. This is an alternative (which will be especially useful in DOS batch files) to responding to the prompts and then indicating "cmf" and then "sjlb.cmf". This option "-cmf" can be used with GEMSIM, TABLO-generated programs and SAGEM.

You can also use **`sj -sti STI-file-name`** under DOS to specify a Stored-input file.

If you include either **`-cmf filename`** or **`-sti filename`** on the command line, you can also specify the name of the LOG file on the command line with either of the options

```
-log  log-file-name
-lon  log-file-name
```

With “-lon” the output goes only to the log file whereas with “-log” the output also goes to the screen. [But the “-log” or “-lon” options cannot be used unless either “-cmf” or “-sti” is also present in the command.]

For example, the command

```
sltoht  -sti  sim1.sti  -lon  sltoht1.log
```

will take inputs from Stored-input file SIM1.STI and will direct output (only) to SLTOHT1.LOG. [This is an alternative to input and output redirection using “<” and “>” under DOS.]

We think that the **`-sti`** option may be more robust than input redirection “<” under Windows NT in some circumstances.

The command line options described above are only available under DOS. At present they are not available on the other platforms (for example, unix) running GEMPACK (though it may be possible to remedy this in the future).

7.4 Stored-input Files, Input Redirection and LOG files

Sometimes when running a "batch" job from a Stored-input (STI) file you needed to choose whether the output went to both the terminal and a log file (B) or to just a log file (L). And sometimes you didn't need the B or L and ended up with a Logfile called "B". We have found a small bug (which was present in Release 5.2) which goes most of the way to explaining (and fixing) the great Log B/L confusion.

With the fix made in Release 6.0, you can use the same Stored-input file by choosing the STI option in the GEMPACK program or by using input redirection "<". This will work without trouble provided **BAT** is the first line of the STI file. If you then choose option LOG in the STI file, the line following LOG MUST be either B or L. For example, your STI file could start with the lines,

```
BAT
LOG
B
SJLB.LOG
```

If you use option SIF to create the STI file in the first place, the BAT is added automatically. However, if you are creating the Stored-input file by hand, make sure the first line is **BAT,** then if later in the file you select LOG, follow it with either B or L.

The idea behind BAT is to tell the program to stop if some input which the program judges to be incorrect is entered. If you are using file redirection without BAT, the program does not know it is running in "batch" mode and will go gaily ahead even after wrong input. This may have unpleasant results like deleting files you wanted to save by mistake. If you place BAT at the top of any STI file, the program will stop after any incorrect input.

Note that, in this connection, “`-sti <STI-file-name>`” on the command line (see section 7.3) behaves exactly as if you had responded “`sti`” and then given the name of the Stored-input file.

CHAPTER 8

8. Miscellaneous Changes from Release 5.2 to Release 6.0

8.1 Speeding Up Simulations

(a) **Automatic Accuracy Simulations**

This refers to simulations which are carried out to user-specified accuracy, as described in section 12.3.

When a subinterval is repeated (because insufficient accuracy was obtained), in the first step of the repeated subinterval, the software uses all the results from the formulas, equations and LU decomposition from the previous time this subinterval was solved. This greatly speeds up this first step of the repeated subinterval.

(b) **Simulations using MA28**

With Release 5.2, when MA28 is used to solve a simulation, various work files are created and read later. This made MA28 slower than MA48 in Release 5.2, even if they took the same times for the LU decomposition. [See section 12.1 for information about MA28 and MA48.]

With Release 6.0, these work files are not used with MA28. Rather processing is done in memory (which speeds things up) in essentially the same way as when MA48 is used. This should mean that the total solution time is directly related to the LU decomposition time.

For example, if LU decomposition with a particular model is quicker with MA28 than with MA48, the total solve time using MA28 should be less than using MA48, and vice versa.

8.2 Transferring Long Names when executing Write statements

When writing an array to a Header Array file, if the long name to use is not specified in the TABLO Input file, TABLO-generated programs and GEMSIM transfer the long name (if not all blanks) from when this same data was read. The intention here is to make it easier for modellers to preserve long names on files when doing data organisation tasks. [We are grateful to Mark Horridge for suggesting this.]

More precisely, when these programs write an array to a Header Array file, the long name written is as set out in the rules below.

(i) If the long name is specified in the TABLO Input file, this long name is used. For example, for the WRITE statement,

```
WRITE V5BAS to file xx header "YYYY" longname "Y Matrix" ;
```

the long name is as given "Y Matrix".

(ii) Otherwise, if exactly the same part of the coefficient has been read from a Header Array file earlier in this TABLO Input file, the long name read in from the data file is used. For example,

```
READ V5BAS from file yy header "ABCD" ;
WRITE V5BAS to file  xx header "YYYY"  ;
```

the long name for the written array at header "YYYY" is the same as the long name for the array on file yy at header "ABCD".

Here the "part of the coefficient" means the sets and elements specified in the write statement. For example,

```
READ V5BAS from file yy header "ABCD" ;
WRITE (all,c,COM) V5BAS(c,"imp") to file xx header "V5BS" ;
```

the long name read at the earlier READ statement will NOT be used in the WRITE statement since, for the array V5BAS at header "ABCD", all of V5BAS is read but only the "imp" part of V5BAS is being written.

If there is no long name on the read data file(s), option (iii) is tried.

(iii) Otherwise the labelling information (given between #'s on the TABLO Input file) for the Coefficient being written is used as the basis for the long name. For example,

```
COEFFICIENT(all,c,COM)(all,s,SOURCE) V5BAS(c,s) #Other Demands# ;
WRITE V5BAS to file  xx header "YYYY"  ;
```

the labelling information "Other demands" becomes the long name written for the array with header "YYYY".

If all of the coefficient is being written, the long name is just this labelling information. If only part of the coefficient is being written, the long name used is the coefficient name followed by sets or elements as arguments (saying which part is being written) followed by the labelling information for the coefficient.

For example, if the write statement is

```
WRITE (all,c,MARGCOM) V5BAS(c,"dom") to file xx header "V5BS" ;
```

the long name will written will be the coefficient name V5BAS(MARGCOM,"dom") followed by the coefficient labelling information for coefficient V5BAS, so the long name is

```
V5BAS(MARGCOM,"dom")  Other Demands
```

8.3 Automatic accuracy with models which read data from the terminal

In Release 5.2, to use automatic accuracy (see section 12.3) with such models, the simulation was required to start from existing Equations and BCV files. This restriction is removed with Release 6.0.

8.4 Where an ASSERTION Fails

GEMSIM and TABLO-generated programs now give full details of the arguments (element names if available or element numbers) whenever an ASSERTION fails. [ASSERTIONs are introduced in section 11.4.]

8.5 Scaling equations

A new statement on Command files (including for SAGEM) is

```
scale equations = yes|NO ;    ! default is no

! "yes" means scale the equations - this is done automatically, no
!  user input is required.
```

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.**

8.6 Shocking a variable with no components exogenous

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

```
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 GEMPACK; GTAP is taken as the example in that paper. See also section 15.2.] 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.

8.7 Saving BCV Files and Using Them

If you wish to save a BCV file, you must now include a "`bcv file = …;` " statement in your Command file (see section 7.2.1).

If you wish to save Equations and BCV files to reuse in a subsequent simulation, you need to include both "`equations file = … ;`" and "`BCV file = … ;`" statements, and you should give the files the same names.

8.7.1 Using Existing Equations and BCV Files Not Supported in Future Releases ?

This is to let you know that we are seriously considering no longer supporting (in the next release of GEMPACK after Release 6.0) the use of existing Equations and BCV files for starting a simulation. Although the use of these can speed up simulations a little, we doubt that the extra complexity that it is causing in our code is sufficient to outweigh the increased risk of bugs in an endeavour to continue supporting this feature.

Accordingly we advise you to being removing the use of these files from your existing Command files whenever you come across it.

If you have strong views that we should retain this feature in future releases, we will be pleased to hear from you.

8.7.2 TABLO Input file written on the Auxiliary files

In Release 6.0, the TABLO Input file for your model is written to the Auxiliary Table file (.AXT for a TABLO-generated program, .GST for GEMSIM) by default. It is written in binary format but there are special programs which can convert it back to a text file. Please contact us at the Impact Project if you need to use this facility to recover a lost TABLO Input file and we will send you the program TEXTBI.EXE which is used to convert back to the text format of the TABLO Input file.

If, for reasons of confidentiality, you do not wish to send out your TABLO Input file as part of the model, you can turn off this default. At the first option screen in TABLO or at the top of your condensation Stored Input file, select the option

NTX Don't store the TAB file on Auxiliary file

then continue as usual with the TABLO Input.

8.8 Checking the Closure and Shocks

If a Command file contains statements specifying the closure and shocks, TABLO-generated programs and GEMSIM now 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, 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
nas = yes ; ! 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).[11] [Of course you must comment these out when you really want the simulation to run.]

[11] Under Release 5.2 of GEMPACK, the program would have told you that there are no actions, hence nothing to do. Release 6.0 programs regard checking the closure and shocks as actions worth doing.

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.

8.9 Additional Shock (ashock) Statements for Dynamic Models

If you work with recursively dynamic models such as MONASH and MEGABARE, 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 6.6 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.

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. With Release 6.0 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.[12]

Syntax and Semantics for ashock statements

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

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

You must not put two ashock 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.

Although ashock statements are valid for all simulations, we recommend that you only use them if you are carrrying out a set of policy simulations with a dynamic model.

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

CHAPTER 9

9. WinGEM and RunGEM on Windows PCs

WinGEM is the Windows interface to GEMPACK introduced with Release 5.2 of GEMPACK. It is aimed at improving the efficiency and productivity of modellers (those who build new models or modify existing models) by providing tools in the familiar Windows environment.

RunGEM is a windows program (new for Release 6.0 of GEMPACK) which is aimed at **model users**, that is, those who carry out simulations with existing models. RunGEM users do not require a GEMPACK licence to solve medium-sized models.

WinGEM and RunGEM are only available on PCs running under Microsoft Windows 3.1, Windows 95 or Windows NT.

The features of WinGEM are described in section 9.1 below. The only major change since Release 5.2 is the inclusion of Mark Horridge's ViewSOL program (for looking at simulation results - an alternative to GEMPIE) in WinGEM.

We expect that new GEMPACK users working on PCs will want to use WinGEM. A detailed tutorial introduction to GEMPACK via WinGEM is available - see section 9.2 below.

The features of RunGEM are described in section 9.3 below.

9.1 WinGEM - the Windows Interface to GEMPACK

In order to allow modellers to carry out modelling tasks in a Windows environment rather than the older DOS-screen environment, **WinGEM** or "**GEMPACK for Windows**" is a Windows interface to GEMPACK. Most of the common modelling tasks can be carried out more easily in WinGEM.

When you use WinGEM, help is available on line in the usual Windows manner. Note that WinGEM runs the usual GEMPACK DOS programs in the background and provides a transparent interface to the results of these modelling tasks. It allows you to take advantage of standard Windows features such as choosing files to open or run from a standard Windows open box.

WinGEM also provides an easy transition from one task to another (for example, from TABLO -> carrying out a simulation -> running GEMPIE to print the results).

WinGEM also carries out checks whenever it can and warns you if these are not satisfied.

When you use WinGEM, you can instruct it to start one task (for example running a simulation) and then carry out other tasks (in different windows) while you are waiting for that task to finish. [WinGEM notifies you when each task has completed.]

If you have GEMPACK and WinGEM installed on your machine, you can choose to carry out modelling tasks in a DOS box (as previously) or via WinGEM. We expect that WinGEM will improve the productivity of both experienced and inexperienced modellers.

Note that WinGEM can only be run successfully if your GEMPACK programs are from **Release 5.2** (or later) of GEMPACK. Either the source code or the executable-image version of GEMPACK can be used.

WinGEM, ViewHAR and ViewSOL need to be installed separately (as Windows programs) in addition to the installation of GEMPACK (Release 5.2 or later) on these machines.

WinGEM, ViewHAR and ViewSOL are available via the World-wide Web at address

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

Installation instructions are available there.

9.1.1 ViewHAR for Looking at or Modifying Data on a Header Array File

ViewHAR is a Windows program written by our colleague Mark Horridge. It allows you to examine the data on a Header Array file in a Windows fashion. It also allows you to print selected data and to look at data summed across various dimensions. When you use ViewHAR you are always looking at a matrix or vector formed from the possibly 3-dimensional (or higher) array on the file.

If the array you are looking at has set and element labelling information (see chapter 5 above), this will be available through ViewHAR.

ViewHAR has on-line help.

Data on the screen can be exported to the Clipboard from where it is possible to paste the data directly into a spreadsheet program such as Excel.

The most recent versions of ViewHAR can be used to modify the data on a Header Array file.

9.1.2 ViewSOL for Looking at Simulation results on a Solution file

View SOL is a Windows program again written by Mark Horridge similar to ViewHAR but this time for looking at results on GEMPACK Solution files (that is, ViewSOL is an alternative to GEMPIE). In fact it can be used with more than one Solution file so you can compare different simulation. ViewSOL provides various ways of grouping variables, for example, by name or set size or all results related to a particular element name, for example all the "food" results.

ViewSOL has on-line help.

Results on the screen can be exported to the Clipboard from where it is possible to paste the table of results directly into a spreadsheet program such as Excel.

9.2 Getting Started with GEMPACK and WinGEM

Section 2 of the GEMPACK document GPD-8 *Getting Started with GEMPACK: Hands-on Examples* contains a step-by-step introduction to GEMPACK via WinGEM.[13]

[13] This was previously Appendix B of the Release 5.2 version of GPD-4, but we have decided to make it available as part of a separate document.

9.3 RunGEM for Model Users

RunGEM is a windows program (new for Release 6.0 of GEMPACK) which is aimed at **model users**, that is, those who carry out simulations with existing models. RunGEM users do not require a GEMPACK licence to solve medium-sized models.

RunGEM makes it easy for users to conduct a variety of model simulations with any model which has been implemented using GEMPACK. RunGEM provides an easy-to-use windows environment in which users can

- choose the model they are working with, eg GTAP or ORANIG.
- choose any group of input data files for the model; these are the starting points for your simulation.
- work with a standard closure for the model or can modify this closure.
- set values for the exogenous variables interactively, or read them from prepared files.
- choose names for the output files from the simulation (including the Solution file and the updated data files).
- choose among several methods to solve the model.
- view simulation results on-screen, print them, or export them to other programs.
- view, copy or print either the initial or the post-simulation model database.

We expect that most users of RunGEM will work with just one model and, much of the time with just one set of pre-simulation data files. Then, in order to carry out a new simulation, they just need to specify the closure and shocks and then press the Solve button. RunGEM provides an interface for all these tasks. For example, when specifying shocks, users can use the mouse to

- select from a list of all variables of the model those they wish to shock,
- specify the numerical value of the shock,
- specify specific components to shock. In this case RunGEM tells the user the sets over which the arguments of the variable range and provides lists of the elements of these sets. Users can click on the relevant element names.

Users of RunGEM need have no prior knowledge of GEMPACK or the syntax of Command files. They select all the ingredients for a simulation using the windows interface and then click on the Solve button. RunGEM does the rest.

When a simulation has been carried out by RunGEM, the simulation results are available on the screen via windows essentially the same as those used by ViewSOL. RunGEM also automatically makes the original and post-simulation data files available via menu options - users do not have to know or remember the names of the actual files.

If the model in question produces a Display file or WRITEs to another output file, RunGEM not only makes these files available but also automatically produces the versions of these files based on the updated data. Thus, if the model in question produces a Display or WRITE file showing GDP, RunGEM automatically makes available the pre-simulation and post-simulation values. We expect that this feature will make RunGEM an attractive way of carrying out simulations even for experienced GEMPACK and WinGEM users.

Whenever WinGEM carries out a simulation, it produces a Command file which can be reused outside of RunGEM. Experienced users of RunGEM can easily specify the names of the output files, and place them anywhere on their hard disk. However, inexperienced users do not need to be concerned with file names at all.

WinGEM is based on RunGTAP which is a similar interface to the GTAP model. We are grateful to Mark Horridge, who wrote RunGTAP, and to Tom Hertel (Director, Center for Global Trade Analysis, Purdue University), who commissioned RunGTAP and helped plan its properties, for permission to incorporate many of the features from RunGTAP into RunGEM.

9.3.1 Preparing a Model for Use with RunGEM

Modellers with a source-code or executable-image version of Release 6.0 of GEMPACK can easily prepare a model for use with RunGEM. Basically, RunGEM requires either

- the TABLO-generated program and associated Auxiliary files for the model, or
- the GEMSIM Auxiliary files for the model. (This is only appropriate if the RunGEM user also has access to GEMSIM, with a suitable GEMPACK licence.)

These must have been prepared using Release 6.0 (or later) of GEMPACK.

Also required are

- one or more sets of pre-simulation data files for the model.
- one or more closure files. (A closure file is that part of a Command file specifying the closure.)[14]
- the Model Information file produced when TABLO is run. (This file, which has suffix .MIN, is always produced when the Release 6.0 version of TABLO writes a TABLO-generated program or output for GEMSIM.)

Details can be found in the on-line help file which comes with RunGEM.

[14] At present RunGEM requires that this be in the form of the list of exogenous variables followed by a "rest endogenous ; " statement. This requirement may be relaxed for later versions of RunGEM.

CHAPTER 10

10. TABLO-like Statements in Command Files

This chapter describes features which were new for Release 5.2. These features have not changed in Release 6.0.

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 or DISPLAYs.

These extra statements are carried out **as if they were appended to the end of the original TABLO Input file**. 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 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)".]
* 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 2.3.1 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) ;
```

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.

10.1 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.

10.2 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). 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.

CHAPTER 11

11. Release 5.2 Enhancements to TABLO Input Files

This chapter describes features which were new for Release 5.2. These features have not changed in Release 6.0.

Several additional new statements are now allowed in TABLO Input files. Details are given in the sections below.

11.1 Set Complements

There is a new form of **SET** statement used for set complements, namely

```
SET  <new-setname>  =  <bigset>  -  <smallset>  ;
```

Here <bigset> and <smallset> must have already been defined in earlier SET statements, and <smallset> must have been declared as a SUBSET of <bigset>.

This statement means that the elements of <new-setname> are all the elements in <bigset> which are not in <smallset>.

> For example, if COM is the set of all commodities and MARCOM is the set of all margins commodities, then the statement
>
> ```
> SET NONMARCOM = COM - MARCOM ;
> ```
>
> says that NONMARCOM consists of all commodities which are not in MARCOM. [It is no longer necessary to list or read the elements of NONMARCOM once those of COM and MARCOM have been specified.] .

As well as defining a new set, a set complement statement automatically generates the obvious **SUBSET** statement, namely

```
SUBSET <new-setname> IS SUBSET of <bigset> ;
```

[In the example above, NONMARCOM is automatically a SUBSET of COM.]

If both <smallset> and <bigset> have **fixed elements** (that is, those defined explicitly in the TABLO Input file), then the complement set <new-setname> has fixed elements.

For example, with the statements

```
SET COM (c1,c2,c3,c4,c5) ;
SET MARCOM (c3, c4) ;
SUBSET MARCOM is subset of COM ;
SET NONMARCOM = COM - MARCOM ;
```

the set NONMARCOM has fixed elements c1,c2,c5.

If <bigset> has run-time elements (read from a file), then <new-setname> also inherits the appropriate run-time elements.

If <smallset> and <bigset> have fixed sizes, so does <new-setname>. Otherwise <new-setname> has MAXIMUM SIZE the same as that for <bigset>.

Intertemporal sets are not allowed in set complement statements. That is, <bigset> and <smallset> are not allowed to be intertemporal sets and the set qualifier (INTERTEMPORAL) is not allowed in a set complement statement.

Set complements can be used in **XSET extra statements** in Command files (see chapter 10).

11.2 Sets Whose Elements Depend on Data

Sets can be defined in TABLO Input files in ways which depend on data, as in for example,

```
SET SPCOM = (all,c,COM: TOTX(c) > TOTY(c)/5 ) ;
```

which says that the set SPCOM consists of all commodities c in COM for which TOTX(c) is greater than one-fifth of TOTY(c). (This set is defined at the first stage of a simulation using initial values of TOTX and TOTY read in or calculated by initial formulas. It will not change at subsequent steps for updated values of TOTX and TOTY).

The required syntax is

```
SET  <new-set>  =  (All, <index> , <old-set> : <condition>  ) ;
```

which says that <new-set> consists of the elements of <old-set> which satisfy the given condition.

Such a set definition automatically implies the obvious SUBSET statement (namely that <new-set> is a subset of <old-set>).

If the old set <old-set> has element names defined, then names for the elements of the new set <new-set> are inherited at run time.

Only one quantifier (ALL,<index>,<set>) is allowed. The condition follows the colon ':' in this quantifier. [The condition can contain operators like AND, OR.]

Note that (unlike for Release 5.2), the resulting set can be empty (since empty sets are allowed in Release 6.0).

At present, set statements of the kind described in this section which depend on data cannot be included as "extra" statements in Command files (see chapter 2 above).

11.3 Mappings Between Sets

Mappings between sets can be defined via new statements following the syntax

```
MAPPING  <set-mapping>  FROM  <set1>  TO  <set2>  ;
```

Here <set1> and <set2> must have already been declared as sets. The mapping called <set-mapping> is a function which takes an element of set <set1> as input and produces as output an element of set <set2>.

Example 1 - Unique Production

Suppose each commodity in the set COM is produced by only one industry in the set IND. We can declare a mapping called PRODUCER from COM to IND as follows.

```
MAPPING PRODUCER from COM to IND ;
```

Then PRODUCER("Com1") would be an element of set IND, namely the industry in IND which produced "Com1". This is an example of a one-to-one mapping if each industry produces only one commodity.

Below we give an example of a many-to-one mapping. You can not use as a mapping a many-to-many relation or a one-to-many relation where one element of the first set corresponds to more than one

element of the second set. (In the example above this would be where a commodity is produced by more than one industry.)

Example 2 - Aggregating Data

Imagine a data manipulation TABLO Input file which is aggregating some data set with commodities in set COM to an aggregated set of AGGCOM commodities.[15] The following statement could be useful.

```
MAPPING COMTOAGGCOM from COM to AGGCOM ;
```

With this, for each "c" in COM, COMTOAGGCOM(c) is the aggregated commodity to which "c" is mapped. The formula for aggregating domestic household consumption could then be written as follows.

```
COEFFICIENT (all,aggc,AGGCOM) AGGDHOUS(aggc) ;
COEFFICIENT (all,c,COM) DHOUS(c) ;
FORMULA (all,aggc,AGGCOM)
AGGDHOUS(aggc) = SUM(c,COM: COMTOAGGCOM(c) EQ aggc, DHOUS(c) );
```

[This uses an "index expression comparison" (see section 11.5.2 below) to compare COMTOAGGCOM(c) with "aggc".]

Example 3 - Aggregating Simulation Results

With COM, AGGCOM and the mapping COMTOAGGCOM as in Example 2 above, it is a simple matter to report simulation results for the aggregated commodities. For example, consider the variable **xhous(c)** which reports the percentage change in household consumption of commodity c in COM. Suppose that you wished to aggregate these (and possibly other similar results) to report the percentage change in household consumption of each of the aggregated commodities. Then you might use the mapping COMTOAGGCOM and the TABLO statements:

```
VARIABLE (all,c,COM) xhous(c) ;
VARIABLE (all,aggc,AGGCOM) agg_xhous(aggc) ;
EQUATION E_agg_xhous (all,aggc,AGGCOM)
SUM(c,COM: COMTOAGGCOM(c) EQ aggc, DVHOUS(c)) * agg_xhous(aggc) =
SUM(c,COM: COMTOAGGCOM(c) EQ aggc, DHOUS(c) * xhous(c) );
```

Here the DVHOUS(c) (as in Example 2 above) are suitable weights to use to aggregate the xhous results.[16]

The procedure in Example 3 is perfectly general and can be used with any model to report aggregated results (calculated while the rest of the model solves); this is more reliable and accurate than trying to aggregate percentage changes after the simulation has been run. [The only slightly tricky thing is sometimes finding suitable weights to use.]

---

[15] As a very simple example, perhaps COM has the 4 elements wool, wheat, banking, hotels and AGGCOM has the two aggregated commodities agriculture (the aggregation of wool and wheat) and services (the aggregation of banking and hotels).

[16] Using the commodities in the previous footnote, note that the equation says, for example, that
[DVHOUS("wool")+DVHOUS("wheat")]*agg_xhous("agriculture") =
DVHOUS("wool")*xhous("wool") + DVHOUS("wheat")*xhous("wheat").
You can see that agg_xhous("agriculture") is a suitably weighted average of xhous("wool") and xhous("wheat").

Of course MAPPINGs must be defined. That is, the TABLO Input file must give a way of specifying <set-mapping>(s) for all s in <set1>. And the values of <set-mapping>(s) must be in the set <set2> for each s in <set1>.[17]

11.3.1 Defining Set Mapping Values

Values for set mappings can be established in four ways, namely

- **by reading the element NAMEs in <set2> where each element of <set1> is mapped**.
  An example is

  ```
  READ (BY_ELEMENTS) COMTOAGGCOM from File setinfo ;
  ```

  The qualifier "(BY_ELEMENTS)" is required. The file *setinfo* can be a text file or a Header Array file (in which case a HEADER must also be specified). The data on the file at the relevant place must be CHARACTER data giving, for each c in COM, an element name in AGGCOM.

- **by formula(s) assigning the element NAMEs** in <set2> to which the different elements of <set1> are mapped. An example is

  ```
  FORMULA COMTOAGGCOM("C10TCF") = "AC5Manuf" ;
  ```

  where the RHS is an element of AGGCOM. This uses element NAMEs on the RHS.

- **by reading the element NUMBERs in <set2> each element of <set1> is mapped**.
  An example is[18]

  ```
  READ COMTOAGGCOM from File setinfo ;
  ```

  The file *setinfo* can be a text file or a Header Array file (in which case a HEADER must also be specified). The data on the file at the relevant place must be INTEGER data giving, for each c in COM, an element number in AGGCOM. [Note that the qualifier "(BY_ELEMENTS)" is not present when integer data is read; this qualifier indicates character data.]

- **by formula(s) assigning the element NUMBERs** in <set2> to which each element of <set1> is mapped. An example is

  ```
  FORMULA  COMTOAGGCOM("C10TCF") = 5 ;
  ```

  where the RHS is the element number in AGGCOM. This uses element NUMBERs on the RHS. [Indeed, in this case, the RHS can be any expression allowed on the RHS of FORMULAs whose LHS is an Integer Coefficient - see section 4.4.10 of GPD-2 for details. For example, the RHS could be ICOEF+2 in the example above if ICOEF had been declared as an Integer Coefficient.]

---

[17] Set mappings do not need to be "onto" in the mathematical sense. Consider a mapping from SET1 to SET2; it need not be the case that every element in SET2 is mapped to by some element in SET1. [Of course, for data aggregation, you may well want this, but the onus is on you to check that it happens.]

[18] The Release 5.2 documentation about set mappings inadvertently omitted information about reading (and writing - see later in this section) set mappings as integers.

Example - Aggregating Data or Results

Consider the following statements

```
SET  COM (wool, wheat, banking, hotels) ;
SET  AGGCOM (agriculture, services) ;
MAPPING COMTOAGGCOM from COM to AGGCOM ;
```

The mapping COMTOAGGCOM could be defined in one of the following three ways.

(1) Via the following formulas with element NAMEs on the RHS

```
Formula  COMTOAGGCOM("wool") = "agriculture" ;
Formula  COMTOAGGCOM("wheat") = "agriculture" ;
Formula  COMTOAGGCOM("banking") = "services" ;
Formula  COMTOAGGCOM("hotels") = "services" ;
```

(2) Via the following formulas with element NUMBERs on the RHS

```
Formula  COMTOAGGCOM("wool") = 1 ;
Formula  COMTOAGGCOM("wheat") = 1 ;
Formula  COMTOAGGCOM("banking") = 2 ;
Formula  COMTOAGGCOM("hotels") = 2 ;
```

(3) Via the following file and read statements

```
File (text)  setinfo ;
Read (BY_ELEMENTS) COMTOAGGCOM from file setinfo ;
```

In this case, the text file SETINFO should have the following array on it.

```
4 strings length 12 ;
agriculture
agriculture
services
services
```

(4) Via the following file and read statements[19]

```
File (text)  setinfo ;
Read COMTOAGGCOM from file setinfo ;
```

In this case, the text file SETINFO should have the following array on it.

```
4 integer ;
1  1  2  2
```

Note the general form of defining statements for COMTOAGGCOM:
COMTOAGGCOM("element from COM") is equal to either an "element of set AGGCOM" or else to an element number of set AGGCOM.

When you assign values to a set mapping, GEMSIM and TABLO-generated programs check that the values are sensible. For example, with COM, AGGCOM and COMTOAGGCOM as above, the formulas

```
COMTOAGGCOM(“wool”) = “steel” ;
COMTOAGGCOM(“wool”) = 15 ;
```

would lead to errors at run time since the “steel” is not an element of AGGCOM and AGGCOM does not have 15 elements.

Note however that, as far as GEMPACK is concerned, set mappings do not need to be “onto” in the mathematical sense. Consider a mapping from SET1 to SET2; it need not be the case that every

[19] In examples (3) and (4) here, a Header Array file could be used instead of a text file.

element in SET2 is mapped to by some element in SET1. [Of course, for data aggregation, you may well want this; if so, the onus is on you to check that it happens.][20]

11.3.2 Using Set Mappings

Set MAPPINGs can be used in arguments of VARIABLEs and COEFFICIENTs.

For example, continuing on the aggregation example, we could write

```
COEFFICIENT (all,c,COM) SHARE(c)
# Share of com c in aggregate com # ;
FORMULA (all,c,COM)
SHARE(c)= DHOUS(c) / AGGDHOUS( COMTOAGGCOM(c)) ;
```

We can write `AGGDHOUS(COMTOAGGCOM(c))`
since AGGDHOUS requires one argument which is in the set AGGCOM and "COMTOAGGCOM(c)" is in AGGCOM since "c" is in COM.

But it would be an error to write `DHOUS(COMTOAGGCOM(c)) !Wrong`
because DHOUS requires one argument which is in the set COM yet COMTOAGGCOM(c) is an element of the set AGGCOM (so is not in the set COM since AGGCOM is not a subset of COM).

In Release 5.1, the argument of a variable or coefficient could only be either an index or the element of a set. Now it can also be a mapping of an index (or even a mapping of a mapping).

11.3.3 Index Expressions

An index expression is either an index or an expression built up from set mappings applied to an index. Index expressions can be used as the arguments of coefficients or variables, or in index-expression comparisons (see section 11.5).

Set MAPPINGs can **ONLY be used in "index expressions"** - that is, in arguments of VARIABLEs or COEFFICIENTs as in

```
AGGDHOUS(COMTOAGGCOM(c))
```

or in index-expression-comparisons such as

```
COMTOAGGCOM(c) EQ aggc
```

in the aggregation example given earlier.

11.3.4 Other Semantics for Mappings

Several READs and/or FORMULAs can be used to set up the values of each set mapping. But, once a set mapping is used (as in section 11.3.2 above or in an index expression, as in section 11.3.3), its values cannot be changed (that is, it cannot appear in a READ statement or on the LHS of a Formula).

Set mappings can be applied to expressions which themselves involve one or more set mappings. In mathematical parlance, this is **composition** of functions.

---

[20] Maybe a later version of GEMPACK will include a qualifier such as "(ONTO)" so that, if you declare a mapping via a MAPPING(ONTO) statement, the onus will be on the software to check that every element is indeed mapped onto at least once.

For example, suppose that MAP1 and MAP2 are set mappings.

```
MAPPING MAP1 FROM S1 TO S2 ;
MAPPING MAP2 FROM S3 TO S4 ;
```

Since MAP1 maps set S1 into set S2 and MAP2 maps set S3 into set S4, then the expressions

```
MAP1(MAP2(i))
MAP1(MAP2("element"))
```

are legal provided that

- index i is ranging over set S3 or over a subset of S3,
- "element" is an element of set S3,
- S4 is the same as S1 or else S4 is a subset of S1.

[For example, then MAP2(i) is in the set S4 because "i" is in S3 and so MAP2(i) is also in the set S1 since S4=S1 or S4 is a subset of S1; hence MAP1(MAP2(i)) makes sense since the argument "MAP2(i)" of MAP1 is in the set S1, as required.] The resulting expressions are thought of as being in set S2 (since this is where the outer mapping MAP1 maps things).

Set mappings are not allowed in index expressions on the LHS of a formula or of an update. For example, the following is not allowed

```
FORMULA (all,c,COM) C1(COMTOAGGCOM(c)) = C2(c) ;  !Wrong !
```

[If several different "c"s in COM were mapped to the same place in AGGCOM, the results would be ambiguous since it would depend on the order of processing the elements "c" in COM.]

Set mappings are not allowed in the declaration of coefficients or variables. Nor are they allowed in reads, writes or displays.

11.3.5 Writing the Values of Set Mappings

Although Set Mappings are not the same as Integer Coefficients, they can be written as if they were. For example, the statement

```
WRITE COMTOAGGCOM to terminal ;
```

is allowed. The values of the set mapping COMTOAGGCOM will be written as integers (which will show the position number in AGGCOM of COMTOAGGCOM(c) for each c in COM).[21]

[21] The Release 5.2 documentation about set mappings inadvertently omitted information about reading and writing set mappings as integers.

11.4 Assertions in TABLO Input Files

A new type of statement called ASSERTION is allowed in TABLO Input files. These allow modellers to request the software to check conditions they expect to hold. For example, the following ASSERTION statement requests the software to check that no entries in a certain array are negative.

```
ASSERTION # Check no negative values # (all,c,COM) DVHOUS(c) >= 0 ;
```

The syntax is

```
ASSERTION [<qualifiers>] [# message #][<quantifier-list>]<condition>;
```

where <condition> is a logical expression, the optional message between hashes '#' is the message that will be shown (at the terminal or in a LOG file) if the condition is not satisfied, and <quantifier-list> (optional) can be one or more quantifiers. If the condition is not satisfied, the run of GEMSIM or the TABLO-generated program is terminated prematurely (that is, just after checking this assertion) and an error message is given.

An example with 2 quantifiers is

```
ASSERTION # Check values not too large #
(all,c,COM)(all,i,IND) X3(c,i) <= 20 ;
```

Qualifiers (ALWAYS) or (INITIAL) are allowed after the keyword ASSERTION. With qualifier ALWAYS (which is the default), the assertion is checked at every step of a multi-step calculation. With qualifier INITIAL the assertion is checked only on the first step.

We strongly advise you to **include a message** (between #'s) with each assertion. Otherwise, if an assertion is not satisfied, you will not know which assertion it is.

If an assertion is not satisfied, the software tells you the element names (or numbers if names are not available) each time it fails.[22] For example, if the assertion

```
ASSERTION # Check no negative values # (all,c,COM) DVHOUS(c) >= 0 ;
```

fails for commodity “wool” you will see the message

```
%% Assertion ‘Check no negative 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 values' does not hold.
```

If an assertion fails on step two or later of a multi-step calculation, you may have more difficulty in finding out the values of the coefficient(s) in question. To see the values of coefficients involved at step two or later, select option '**DWS**' which causes DISPLAYs and WRITEs to the TERMINAL to be done at all steps (but not writes to other files); see section 5.6 of GPD-2 for more information about option 'DWS'.

You can suppress the testing of assertions by inserting the statement **NAS = yes ;** in your Command file.

[22] This is new for Release 6.0. [Release 5.2 software did not show this.]

11.5 Index-Expression Conditions

11.5.1 Comparing Indices

Conditions for SUMs and IFs can now involve comparisons of indices and index expressions. You can also use index comparisons for conditions in ALLs in the cases where conditions on ALLs are allowed (see GPD-2 section 4.3).

You can test if an index is equal to **EQ** or not equal to **NE** another index, as in the formula:

```
FORMULA (all,c,COM)(all,i,COM)    X(c,i) = IF( c EQ i , Y(i)) ;
```

The expression c EQ i is a comparison of two indices. The IF condition c EQ i is true if the element names of the set elements c and i are the same.

This would set X(i,i) = Y(i) and X(c,i) = 0 when c is not equal to i.

Similarly,

```
FORMULA (all,c,COM) Y(c) = SUM(i,COM : i NE c , X(c,i) ) ;
```

will, for each c in COM, sum all X(c,i) except for X(c,c).

11.5.2 Index Expression Comparison

Index expressions are built up from indices and set MAPPINGs (see section 11.3). For example, COMTOAGGCOM(c) is an index expression.

Conditions of the form

<index-expression-1> <comparison-operator> <index-expression-2>

are allowed with <comparison-operator> replaced by **EQ** (equals) or **NE** (not equal to).

The expression "COMTOAGGCOM(c) EQ aggc" in the formula

```
AGGDHOUS(aggc) = SUM(c,COM: COMTOAGGCOM(c) EQ aggc, DHOUS(c) ) ;
```

is an **index-expression comparison.**

If the index expressions are in intertemporal sets, then <comparison-operator> can be replaced by any of:

LE	(less than or equal to)
LT	(less than)
GE	(greater than or equal to)
GT	(greater than)
EQ	(equal to)
NE	(not equal to).

In the general form of the condition above, the set in which <index-expression-1> lies must be either equal to, or else a SUBSET of, the set in which <index-expression-2> lies, or vice versa. [An index by itself lies in the set over which it ranges. When a set MAPPING from <set1> to <set2> is applied to an index ranging over <set1> or a subset of <set1>, the resulting expression is in the set <set2>.]

> For example you cannot compare c with i when c belongs to the set of commodities COM and i belongs to the set of industries IND. But using the mapping PRODUCER defined:
>
> ```
> MAPPING PRODUCER from COM to IND ;
> ```
>
> you can compare the mapping of c, PRODUCER(c), with industry i as in "PRODUCER(c) EQ i" since PRODUCER(c) ranges over the set of industries IND.

When indices are ranging over intertemporal sets, **index offsets** can be used. For example,

```
MAP1(t+2)-3
```

is legal if index "t" is ranging over intertemporal set S1 and MAP1 is a MAPPING from S1 to an intertemporal set S2.

11.6 $POS Function to Indicate Position Number in a Set

The function $POS is used to determine the position number in a set from the index or element name. The function $POS can be used in the following three ways.

- **$POS(<index>)** indicates the position number of <index> in the set over which this index is ranging.

  For example, suppose that COM is the set (c1-c5). The formula

  ```
  FORMULA (all,c,COM) X(c) = $POS(c) ;
  ```

  puts X("c1") equal to 1, X("c2") = 2, X("c3") = 3, X("c4") = 4 and X("c5") = 5.

- **$POS(<index>,<set2>)** indicates the position of <index> in the set <set2>. In this case, <index> must be ranging over a set which is a subset of <set2>.

  For example, consider COM as above and suppose that MARCOM is the set (c3,c4) and that MARCOM has been declared as a subset of COM. The formula

  ```
  FORMULA (all,mc,MARCOM) X(mc) = $POS(mc,COM) ;
  ```

  puts X("c3") equal to 3 and X("c4") = 4 (their position numbers in COM). But the formula

  ```
  FORMULA (all,mc,MARCOM) X(mc) = $POS(mc) ;
  ```

  puts X("c3") equal to 1 and X("c4") = 2 (their position numbers in MARCOM).

- **$POS(<element>,<set>)** is also allowed. This gives the position number of the element <element> in the set <set>.

  For example, with the sets above, $POS("c3",MARCOM) = 1 and $POS("c3",COM) = 3. .

In fact the first argument of $POS can be an index expression. For example, $POS(COMTOAGGCOM(c)) and $POS(COMTOAGGCOM(c),COM) are legal if index "c" is ranging over set COM (or over a subset of it).

Example from ORANI-F

In the TABLO Input file for ORANI-F [the file ORANIF.TAB supplied with GEMPACK and documented in Horridge, Parmenter and Pearson (1993)], the Coefficient ORD(y) for "y" in YEARS is no longer needed; every occurrence of ORD(y) could be replaced by

$POS(y,YEARS)

[or by $POS(y) whenever index 'y' is ranging over the set YEARS].

11.7 Writing Updated Values from FORMULA(INITIAL)s

Previously, when the initial (that is, pre-simulation) values of a coefficient were assigned via a FORMULA(INITIAL), the post-simulation values of this coefficient were not available. Now they can be written to a file by including appropriate instructions in the FORMULA(INITIAL) statement.

For example, the qualifier "WRITE UPDATED ..." below

```
FORMULA (INITIAL,
    WRITE UPDATED VALUE TO FILE upd_prices
                    HEADER "ABCD" LONGNAME "<words>" )
    (all,c,COM) PHOUS(c) = 1 ;
```

will ensure that the updated (ie post-simulation) values of PHOUS will be written to logical file "upd_prices" at the specified header with the specified long name.

In such a case, the logical file can point to either a Header Array file or to a text file. [If a text file, no header or longname is specified.]

A new FILE qualifier "**FOR_UPDATES**" is provided to declare a logical file which can have updated values written to it. For example, the declaration

```
FILE (FOR_UPDATES) io_updated  #to contain updated prices# ;
```

declares a Header Array file which can have updated values written to it. This qualifier "FOR_UPDATES" is an alternative to the current qualifiers 'OLD' and 'NEW'. [See section 3.5 of GPD-2 for information about the FILE qualifiers OLD/NEW.]

The logical file appearing in the "WRITE UPDATED VALUE TO FILE <file>" must either have been declared using FILE qualifier "FOR_UPDATES" or have been declared as an OLD Header Array file. [An OLD text file is not allowed here.] In the case of an OLD Header Array file, the updated values from the "WRITE UPDATED VALUE TO FILE" statement will appear on the same updated data file as the updated values of coefficients originally read from this file.

Of course the HEADER and LONGNAME can only specified when the file in question is a Header Array file and, in this case, HEADER is required and LONGNAME is optional (the long name will be set all blank if LONGNAME is omitted).

When a logical file is declared with qualifier "FOR_UPDATES", nothing can be read from this file and an updated version will be created after the simulation. [So, in your Command file, a statement "updated file <logical-name> = ... ;" is required to specify the post-simulation version of this file, but a statement "file <logical-name> = ... ;" is not required since no pre-simulation version of this file is expected.]

11.7.1 Saving Updated Values from All FORMULA(INITIAL)s

An option

SUI **S**ave **U**pdated values from formula **I**nitials

in GEMSIM and TABLO-generated programs allows you to save updated values of all coefficients whose initial values are set via FORMULA(INITIAL)s. To use this, put statements

```
sui = yes ;
updated formula initial data = <filename> ;
```

in your Command file. 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 11.7 above) in it will be written to this file.

Example

If you add the following statements to the Command file SJLB.CMF (see Figure 2.3.1 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 12

12. Solving Models

This chapter describes features which were new for Release 5.2.

12.1 New Sparse-Linear-Equation Solving Routines MA48

Until Release 5.1, GEMPACK used the Harwell Laboratory's sparse-linear-equation-solving routines MA28. These routines are still available in Release 5.2 but the default in 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.

12.1.1 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.

12.2 Extrapolation Accuracy Summaries for Data and for Levels Variables

12.2.1 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 currently gives such estimates for each component of each endogenous variable. Such summaries are now 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 12.3 below.

12.2.2 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 now reports the number of figures you can be confident of for the associated levels variable (as well as for the percentage-change variable as before). 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 2.5.3 of GPD-1) now 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) now give a summary for these levels results (as well as for the linearised variables, as before). 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 12.3 below.

12.3 User-specified Accuracy for Simulations

With Release 5.1 of GEMPACK, you had to guess in advance how many steps to take and how many subintervals (see section 5.8 of GPD-2 and also section 12.3.1 below) to use to obtain the accuracy you required. With Release 5.2, you can now 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.

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 specify the accuracy

accuracy figures = <d> ;   ! 4 is the default
        ! e.g. accuracy figures = 5 ;

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

accuracy criterion = DATA|solution|both ;  ! data is the default
        ! e.g. accuracy criterion = both ;
```

If you specify "automatic accuracy = yes ;" then the default is to require at least 80 percent of the extrapolated data results to be accurate to at least 4 figures. [See section 12.2.1 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 of the form

```
subintervals = <number> ;
```

in your Command file.

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 redoes 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 12.2.1 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 12.2.2 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 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 2.2), simulations carried out with user-specified accuracy ensure that the coefficients never stray outside the specified range.

Note that user-specified accuracy is now available without restriction for models which read data from the terminal (see section 8.3).

12.3.1 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.

12.3.2 Incompatibilities

Note that "automatic accuracy = yes ;" is incompatible with option NUD (no final updates - see section 5.6 of GPD-2), with Newton's method (see section 12.4 below), with the SUP and SSL options described in section 12.5 below 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 5.10.2 of GPD-2).

12.3.3 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 [see, for example, Chapter 15 of Press et al (1986)].

12.3.4 Several Subintervals (Revisited)

This section supplements the information in section 5.8 of GPD-2 about the use of several subintervals. Here we are not discussing user-specified accuracy but rather the case where you specify a fixed number of subintervals using the statement

```
subintervals = <number> ;
```

in your Command file (but have no "automatic accuracy = yes ;" statement).

In this case, the subintervals are all of the same length. One advantage of using several subintervals in this way is that, as stated in section 5.8 of GPD-2, you get early warning of unsatisfactory accuracy.

A **second advantage** (not stated in section 5.8 of GPD-2) 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 recently, 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 in GPD-2 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.

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.]

12.4 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 now be used in GEMPACK with levels (or mixed levels/linear) models in certain circumstances. 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
```

To use TABLO to add this $del_newton term to all levels equations you choose option

NWT - Add Newton-correction terms to levels equations.

from the main TABLO menu (the one presented before the CHECK stage). When you run GEMSIM or the TABLO-generated program (after compiling and linking as usual), you must use Euler as the solution method and also set the variable $del_newton exogenous. However there is no need to give the shocks for $del_newton.

To use the Newton method of solution when running GEMSIM or the TABLO-generated program, choose the option NWT at the first option screen or put **NWT = yes ;** in your Command file, then continue with the simulation as usual. 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.

12.4.1 Newton's method and ORANIF

In testing this method using a levels version of ORANIF, difficulties were encountered with zeros in the database and zero divides. Since the Newton-correction term is evaluated as part of an equation, you can not use the ZERODIVIDE defaults available for formulas. In general it is possible to use IF syntax 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.

12.4.2 Convergence

When Newton converges, it does so rapidly - you can check convergence by checking the total of the $del_Newton terms (that is, sum of their absolute values) 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.

12.4.3 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.

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

Newton's Method Is Not Reliable with Mixed Models

Note that we do not recommend the use of Newton's method with mixed models (those whose TABLO Input files contain some levels and some linearised equations). In such a case, TABLO is only able to add the Newton correction terms to the levels equations. We have found that the solution appears to converge but that it can converge to the "wrong" solution in these cases (that is, to a solution which is not a solution of the underlying levels model).

Thus

we do not recommend Newton's method with any models except fully levels ones.

12.5 Options for Saving Solution and Updated Data Files

There are two separate options in GEMSIM and TABLO-generated programs which can be used to save files from the separate multi-step solutions as well as the usual extrapolated results.

SSL Save separate multistep solutions

SUP Save updated data files after multistep solutions

Both of these options are only available on Command files where the relevant syntax is

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 7.3.4 of GPD-1).

> 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 with the subinterval option

SSI - Several subintervals; extrapolate after each

(see section 5.8 of GPD-2 and section 12.3.1 above) or with user-specified accuracy (see section 12.3 above).

CHAPTER 13

13. Other Enhancements for Release 5.2

Some other enhancements which were introduced with Release 5.2 are described below.

13.1 LOG Files and Early Errors

When you select option LOG, the log file now records everything echoed to the screen (even that which occurred before you selected this option - this early output was not recorded in the LOG file in Release 5.1). In particular, if you are using a Command file, the echoing of this file is now captured in the LOG file.

If you are using a Command file and a syntax error occurs while the program is reading this file, the program now produces a LOG file which you can look at to see the error. The program makes up a name for this file (it will be something like "GPXX7.LOG") and tells you this name at the end of the run. This LOG file is created even if you don't have a statement "log file = ... ;" in your Command file.

13.2 Default Output File Names

Several of the utility programs (including SEEHAR and SLTOHT) now usually offer default names for output files they create. For example, if you run SEEHAR to look at the data file SJ.DAT for Stylized Johansen, SEEHAR will suggest the name SJ.SEE for the output (if you are producing SEEHAR's labelled output) or the name SJ.ROW if you select SEEHAR option 'ROW'. We hope that this will make these programs easier to run (especially for inexperienced users). Of course, you can specify other names for these files.

13.3 SEEHAR - New Options

SEEHAR has several new options including options to change the page width **CPW** and change the page length **CPL**, and an option **FNP** which instructs SEEHAR to start a new page only when required for labelled output (the default output from SEEHAR). SEEHAR can also be used to translate a Header Array file to GAMS format (see chapter 16) by selecting option **GMS.**

With labelled output, the previous version of SEEHAR asked you how many figures to put after the decimal point. Now this is an option and the default number of such figures is 6; to change this you need to select option **DEC** and then specify the number of figures you want. Note that this change may mean that existing Stored-input files for SEEHAR will no longer work.

13.4 GEMSIM and TABLO-generated Programs

We collect here some of the enhancements relating to GEMSIM or TABLO-generated programs. Other such enhancements have been described in chapter 11 above.

13.4.1 Updated Versions of Parameter Files No Longer Produced

If all data on a file (Header Array or text) are parameters (that is, are not updated in any multi-step calculation), then updated version of this file are no longer produced by GEMSIM or TABLO-generated programs. Hence you can omit statements specifying the names of updated versions of such files (though an error will not occur if you do specify the names of the updated versions of these files).

However, if some data on one file are updated and other data on this file are not updated, the updated version of the file will contain versions of both sorts of data. [This is so that the file can be used as the starting point for another simulation, if desired.]

13.4.2 Displays of Integer Coefficients

DISPLAYs of integer coefficients are now allowed.

13.4.3 Closure and Shocks in GEMSIM or TABLO-generated Programs

GEMSIM and TABLO-generated programs now often process the closure and shocks earlier than they did in Release 5.1. Precisely, they process the closure as soon as all SET and SUBSET statements (including any "extra" ones - see chapter 10 above) 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.

The closure and shocks are processed earlier to give you quicker 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.

13.4.4 Solution Method and Step Numbers Added to Verbal Description

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. [With Release 6.0, if you use a Command file used to carry out a simulation, its name is also added to the Verbal Description (see section 7.1.1).]

13.4.5 Check that Auxiliary Files are Correct

TABLO-generated programs now check that they 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 forget to compile and link the TABLO-generated program).

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

13.4.6 Reuse of Pivots in Multi-step Simulations

In Release 5.1, during a multi-step simulation, at each step an attempt was made to reuse the pivots from the first step. If reuse failed at say the 4th step, it probably failed at subsequent steps.

With Release 5.2, if reuse fails at one step (say the 4th), the pivots from this step are stored and, at the next step, an attempt is made to reuse these new pivots rather than those from the first step. This makes reuse of pivots more likely to succeed (which can speed up simulations).

13.5 SEENV Program to See the Closure on an Environment File

SEENV is a new utility program to look at the closure on an Environment file, Solution file or LU file. Since all of these files are binary files, they cannot be examined directly, as you would a text file, or transferred to other operating systems. If you have used "swap" statements in setting up your closure, SEENV can be used to look at the resulting closure to check exactly which variables and components are exogenous, and which are endogenous. Variables which have been condensed out of the model do not occur in either list.

After giving the name of the file which contains the environment, and the name of the output text file, you specify the type of output that you would like on the output text file. The two possible types are :

- text suitable for including in a Command file to give the closure, or
- a text file which can be used as a Spreadsheet Mapping file (see section 8.3 of GPD-1).

You must also choose whether you want just the exogenous variables, just the endogenous variables or both types (that is, all the variables in the (condensed) model).

13.5.1 Command file output

If you ask for Command file output and choose **exogenous** variables, the output would be a text file containing the key word "exogenous", followed by a list of exogenous variables, then a semicolon ; to end the exogenous statement, and then the completing statement "rest endogenous ;"

For example, if the exogenous variables are

```
 xx1, xx2 1-5 8-10 13 15, xx3,
```

the text file would contain:

```
exogenous
xx1
xx2
  1 - 5, 8 - 10, 13, 15
xx3
 ;
rest endogenous ;
```

Similarly if you choose **endogenous** output, the form of the output is

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

For **both** exogenous and endogenous, the form of the output is

```
exogenous
<list of exogenous variables>
  ;
endogenous
<list of endogenous variables>
  ;
```

13.5.2 Spreadsheet output

For spreadsheet output, the syntax is that used in Spreadsheet Mapping files (see GPD-1, section 8.3). You can choose to include in the map file, the exogenous variables, the endogenous variables or both types of variables.

For example for the same list of exogenous variables as in the previous example, if you choose **exogenous**, the output spreadsheet map contains the following list of (exogenous) variables:

```
xx1
xx2
  1 - 5, 8 - 10, 13, 15
  ;
xx3
```

The semicolon which ends the list of components is on a separate line (indented one space).

If you choose **endogenous**, the spreadsheet map contains just the endogenous variables and the endogenous components of variables.

If you choose **both**, the spreadsheet map contains a list of all variables in the model, first the exogenous variables and exogenous components of variables, then the endogenous variables and the endogenous components of variables.

13.6 Closure Specified Using Data Files

It is now possible to use data on a text file to specify which components of one or more variables are to be exogenous or endogenous. 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 now allowed.

The "how much data" information must be the same as on a shock file (see section 6.3.1 of GPD-1) 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 ;

13.7 RWHAR - New Options

The current version of RWHAR has a new option to compress the size of the text file written:

CPR Real arrays in compressed form

If this is selected, each number in a real array is written with only 8 figures instead of the usual 12. This is adequate for most machines and reduces the size of the output file by 20-25%. MKHAR can read files containing such compressed real arrays without any user intervention.

A second new option

C51 Convert to Release 5.1 Header Arrays (no set/element labels)

allows you to remove any set and element labelling from a Header Array file. In this case the output from RWHAR is a Header Array file (not a text file, as usual). See section 5.4 above for more details about this option.

13.8 SUMEQ and the Homogeneity of Models

The program SUMEQ can be used to check the homogeneity of models. This is described briefly in section 8.1.2 of GPD-1. We include a detailed example below.

A modified version of SUMEQ (version 1.7, April 1996) distributed with Release 5.2 makes this testing a little easier. When reporting row sums, SUMEQ now suppresses the details of all rows whose sum is less than 0.000002. It also reports the names of the equation blocks (that is the EQUATION name from the TABLO Input file of the model) before each group of rows. This makes it much easier to identify equations which are not homogeneous.

Suppose we have a linearised equation

```
A*p(1) + B*x(1) + C*p(2) + D*x(2) + ... = 0.
```

If we take the nominal variables p(1),p(2)... as 1 and the real variables x(1),x(2)... as 0, this becomes

```
A  +  0  +  C  +  0  + ... = 0.
```

Hence the row sum of the equations summing over just those columns corresponding to the nominal variables p(1),p(2)... should be zero. This is the basic idea behind using SUMEQ's ability to calculate row sums over selected columns to check homogeneity.

13.8.1 Homogeneity Example with Miniature ORANI

Below we give details for using SUMEQ to check the homogeneity of Miniature ORANI (see sections 3-9 of Dixon et al (1982)). The homogeneity in question is the property of the model that one solution of the model is obtained by increasing all domestic (that is, $Australian) prices and dollar values by 1 per cent, while holding all quantities and all foreign prices and dollar values fixed.

To prepare to use SUMEQ to check this, you first need to produce a map of the variables, using SUMEQ (see section 8.1.1 of GPD-1). For Miniature ORANI as in MO.TAB distributed with GEMPACK, such a map is shown below.

```
VARIABLES
 ---------
  No Name          Size   Cols    Arguments (if any) and Description
-------------------------------------------------------------------------
 1 p_XHOUS           4 1-4       (COM,SOURCE)  household consumption
 2 p_CHOUS           1 5           nominal total household consumption
 3 p_PCOM            4 6-9       (COM,SOURCE)  commodity prices
 4 p_PDOT            2 10-11     (COM)  Unit cost of Armington aggrega
 5 p_XDOT            2 12-13     (COM)  Armington aggregate over sourc
 6 p_U               1 14          Consumption mix ratio (outer nest -
 7 p_PEXP            2 15-16     (COM)  export prices (foreign dollars)
 8 p_PIMP            2 17-18     (COM)  import prices (foreign dollars)
 9 p_XEXP            2 19-20     (COM)  export demands
10 p_FEXP            2 21-22     (COM)  export demand shifters
11 p_Z               2 23-24     (IND)  industry activity
12 p_YCOMIND         4 25-28     (COM,IND)  industry output
13 p_XINTCOM         8 29-36     (COM,SOURCE,IND)  intermediate commod
14 p_XINTFAC         4 37-40     (FAC,IND)  intermediate factor inputs
15 p_PLAB            1 41          wage rate
16 p_PCAP            2 42-43     (IND)  price of capital
17 p_PFAC            4 44-47     (FAC,IND)  price of primary factors
18 p_XINTCOM_CD      4 48-51     (COM,IND) Cobb-Douglas combination of
19 p_PCOM_CD         4 52-55     (COM,IND)  Price index for Cobb-D com
20 p_XINTFAC_CD      2 56-57     (IND) Cobb-Douglas combination of pri
21 p_PFAC_CD         2 58-59     (IND)  Price index for combination of
22 p_V               2 60-61     (COM)  power of export subsidy
23 p_T               2 62-63     (COM)  power of import duty
24 p_XLAB            1 64          total demand for labor
25 p_XCAP            2 65-66     (IND)  industry demand for capital
26 p_M               1 67          total imports (foreign or overseas
27 p_E               1 68          total exports (foreign or overseas
28 p_CPI             1 69          consumer price index
29 p_FWAGE           1 70          wage rate shifter
30 p_CR              1 71          real household consumption
31 p_XIMP            2 72-73     (COM)  import quantities
32 p_INTCOM          8 74-81     (COM,SOURCE,IND)
33 p_INTFAC          4 82-85     (FAC,IND)
34 p_HOUSE           4 86-89     (COM,SOURCE)
35 p_EXPCOM_DOMV     2 90-91     (COM)
36 p_IMPCOM_DOMV     2 92-93     (COM)
37 p_COMPROD         4 94-97     (COM,IND)
38 p_PHI             1 98          exchange rate
39 c_B_A             1 99          Aust dollar change in trade balance
40 c_B_F             1 100         Foreign dollar change in trade bala
41 c_EXPSUB          2 101-102   (COM)
42 c_DUTY            2 103-104   (COM)
```

The percentage change variables whose values increase by 1 per cent (that is, which are $A prices or dollar values) are:

```
p_CHOUS, p_PCOM, p_PDOT, p_PLAB, p_PCAP, p_PFAC,
p_PCOM_CD, p_PFAC_CD, p_CPI, p_INTCOM, p_INTFAC, p_HOUSE,
p_EXPCOM_DOMV, p_IMPCOM_DOMV, p_COMPROD, p_PHI
```

[The other variables involving $A values are the change variables c_B_A, c_EXPSUB and c_DUTY. We ignore these for the present, though you will see below that they need to be considered.]

As can be seen from the map of the variables, these correspond to columns numbered 5-11, 41-47, 52-55, 58-59, 69, 74-98. The responses for running SUMEQ are shown below.

User Input to SUMEQ

```
<carriage-return>       ! Default SUMEQ options
mo.eq4                  ! Equations file
mo.seq                  ! SUMEQ output file
s                       ! info about row/col sums
a                       ! all rows
s                       ! just some columns
46                      ! number of columns
5-11 41-47 52-55 58-59 69 74-98   ! the columns
r                       ! just row sums
f                       ! finish
```

The relevant parts of the resulting SUMEQ output file are shown below.

```
                                                ROW SUM OF      NUMBER OF
     ROW NUMBER           ROW SUM                ABSOLUTES       NONZEROS

NOTE.   Rows in which row sum is zero are not shown.
   Rows in which (row sum)/(sum-of-absolutes) is less
    than 2.000000E-06 are not shown.

Starting equation block 'INTERCOM            '
Starting equation block 'INTERLAB            '
  (etc)
Starting equation block 'IMPORT_LEVELS       '
Starting equation block 'DUTYUP              '
       28              -0.09999996              1.90000010        2
       29              -0.29411763              1.70588243        2
Starting equation block 'MARKCLLAB           '
Starting equation block 'MARKCLCAP           '
  (etc)
Starting equation block 'TRADEBALANCE_A      '
      42               1.00000000              1.00000000        1
Starting equation block 'TRADEBALANCE_F      '
Starting equation block 'REALCONSUMPTION     '
  (etc)
Starting equation block 'MARKCLCOM           '
```

All equations sum to sufficiently close to zero (that is, less than 0.000002) except for those in blocks DUTYUP and TRADEBALANCE_A. These nonzero sums are because of the change variables we ignored above. For example, the linearised version of equation TRADEBALANCE_A is essentially

```
100.0*c_B_A - [PHI*(E*p_E - M*p_M) + (E - M)*PHI*p_PHI] = 0
```

(as you can see by looking in the Information file produced when TABLO processes MO.TAB). In the calculations SUMEQ did, we are putting p_E=p_M=0 and p_PHI=1. Thus the nonzero terms on the left-hand side of the linearised equation amount to -(E-M)*PHI*1; but E=20, M=21 and PHI=1 (as you can check from the data file for MO) so this means that the left-hand side is equal to 1.0, as indicated in the SUMEQ output file. You can see that the correct value for the change variable c_B_A in a homogeneity simulation cannot be easily predicted in advance (depending as it does on the E and M data). Similarly for the other equations DUTYUP which involve $A-related change variables. [Indeed

it is something of an accident that the EXPORTSUBS equation is shown with zero sum - this would not be the case if the pre-simulation value of EXPSUBS were nonzero.]

This shows the usefulness (and limitations) of using SUMEQ to check homogeneity. It is an excellent check when percentage-change variables are involved but requires care when change variables are involved.

In larger models which are condensed, SUMEQ is checking the condensed equations. This means that if one equation block is incorrect (not homogeneous, perhaps because some shares do not add to 1), and if this equation is substituted in other equations, all these other equations will look wrong in the SUMEQ output.

13.9 Displays and Labelled Output from SEEHAR into Spreadsheets

Christian Bach has pointed out that DISPLAYs and labelled output from SEEHAR (this is the default output from SEEHAR) can be easily transferred into a spreadsheet using the "space delimited" option available in Excel (version 5.0 or later) and other spreadsheet software. To import these text files into a spreadsheet, select "space" as the delimiter and tell the spreadsheet to treat multiple spaces as a single separator.

To make the resulting spreadsheet line up better, we have added

No. Name

on the left of lines listing element names in Release 5.2 output, as you can see in the example below (a Display of part of the Stylized Johansen data base).

```
 Next submatrix contains numbers from DVCOMIN(SECT,SECT) part, which
   are positions (1-2, 1-2) of DVCOMIN(SECT,SECT).
     COLUMN                1            2         ROW
     ROW                                          TOTALS
  No. Name              s1           s2
    1 s1              4.000000     2.000000     6.000000
    2 s2              2.000000     6.000000     8.000000
    COLUMN
    TOTALS            6.000000     8.000000    14.000000
```

Note that in Display files and labelled output from SEEHAR, unlike spreadsheet output, matrices with many columns may not fit across the page but be split into two or more parts. To minimise this effect, you can increase the page width using a statement

```
display page width = ... ;
```

in your Command file (for DISPLAYs) or via SEEHAR's option CPW (Change Page Width).

See also chapter 6 for several ways of making tables in spreadsheets using SLTOHT.

CHAPTER 14

14. Changes from Release 5.1 to Release 5.2

This chapter describes some features which were different in Release 5.2 from 5.1.

14.1 Clarification Concerning Shocks on a Command File

This section aims to clarify one point concerning shocks on a Command file which is ambiguous in the previous documentation. 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.

14.1.1 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). [We believe that the above information was not clear from section 6.3.1 of GPD-1 or elsewhere in GPD-1.]

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.]

14.1.2 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 chapter 19.1). We believe that this was a dangerous default since it would apply if you inadvertently left out all shock statements. Accordingly we have 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.

14.2 Increasing Parameters in TABLO-generated Programs and GEMSIM

This section remains relevant for Release 6.0 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**.

14.2.1 Parameters MMNZ and MMNZ1

Parameters MMNZ and MMNZ1 have a slightly different meaning in Release 5.2 than they did in Release 5.1. Now MMNZ usually needs to be twice as large as the number of nonzero entries in the Equations file (twice as large as in Release 5.1). 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. As explained in section 5.13.2 of GPD-2, 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.

14.2.2 Other Parameters

If you have several "**extra**" statements on your Command file (see chapter 10 above), TABLO-generated programs may tell you that you need to increase one or more program parameters besides those listed in section 5.13.2 of GPD-2. For example, if you have more than about 10 "xwrite" statements in your Command file, you may be told that you have to increase the value of program parameter MMNWR.

Besides those mentioned in section 5.13.2 of GPD-2, the parameters you may need to increase are

MMNFL, MMNST, MMNSS, MMNWR, MMNIG, MMEL, DCSRIG

If you have to increase one of these, you will need to edit the TABLO-generated program (usually a '.FOR' file) and then recompile and relink it (usually using "LTG" - see point (2) in section 5.13.2 of GPD-2). An alternative is to reduce the number of "extra" statements on your Command file. [Option CDM in the Code stage of TABLO does not allow you to increase the values of the parameters listed above.]

14.3 Licence Files

We have decided not to require a licence file for the GEMPACK utility programs. For Release 5.2, only TABLO and GEMSIM require access to a GEMPACK licence. [But, for Release 6.0, SAGEM requires a licence, as do TABLO-generated programs in some circumstances (see section 1.2.5 of GPD-1).]

New licence files are required for Release 5.2 of GEMPACK. [The Release 5.2 programs requiring a licence (see above) will not run with Release 5.1 licence files.] Some Release 5.2 licences have expiry dates.

14.4 Clarification Relating to Closures and Solution Files

A 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 ;
```

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

14.5 Parts of Release 5.1 No Longer Supported

14.5.1 TABLO Low Memory Code Options SCS,LIR,LRP

These options (see section 5.10.2 of GPD-2) are no longer available with Release 5.2.

14.5.2 Submatrix Data Files and Set-up Files

Submatrix Data, Set-up files and splitting a model into several submodels (see the Code stage of TABLO) are not supported in Release 5.2. This means that

- programs ASSEQ, AMST, SEESMD, SEEST, SUMSMD [see (4) of section 1.1 of GPD-1] are not part of Release 5.2,
- TABLO Code options SMD and SPL are not available in Release 5.2,

statements "`Submatrix Data file = ... ;`" and "`Set-up file = ... ;`" in Command files are not available in Release 5.2.

14.6 Technical Notes for Programmers

This section is only of relevance to programmers who have used GEMPACK routines in their own programs.

There are a few changes in the calling sequences of routines that you may have used in your programs. The following are worthy of notice.

14.6.1 GPSTRT

The second last argument in this call (usually 'x' in the past) should now be replaced by the name of the program (often a variable denoted by PRNAME). The example below should make this clear.

```
          CHARACTER*6 PRNAME
          PARAMETER(PRNAME='GEMPIE')
C
C.old     CALL GPSTRT(PROGNV,DATES,.TRUE.,.FALSE.,'X',ERRMES)
          CALL GPSTRT(PROGNV,DATES,.TRUE.,.FALSE.,PRNAME,ERRMES)
          IF(ERRMES(1:1).EQ.'E') GOTO 15000
```

In this call, the second last argument must be a CHARACTER string. If it is longer than 8 characters, only the first 8 will be significant.

14.6.2 LSTHDS

This routine has three extra arguments in its call (just before ERRMES). For example, in SEEHAR, the new call is

```
         CALL LSTHDS( LUHAR, MMHEAD, 'MMHEAD', HEAD, 1, NHEAD,
      *    TYPE, NELS, MAXDIM, LNAM, .TRUE., .TRUE., .TRUE.,
C [11/03/96] add CNMSTR and TPSTEL for 'RE' arrays and SETSEP
      *    CNMSTR,TPSTEL,SETSEP,
      *     ERRMES)
```

These extra arguments should be declared via statements like

```
          CHARACTER*(120) CNMSTR(MMHEAD)
         INTEGER TPSTEL(MMHEAD)
         CHARACTER SETSEP
```

where MMHEAD is the dimension of the other arrays such as HEAD(MMHEAD) in the call. Here CNMSTR(1..NHEAD) and TPSTEL(1..NHEAD) are returned. SETSEP should be assigned before the call; usually it is ','. [See the code of LSTHDS for more details.]

CHAPTER 15

15. 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.

15.1 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.

Associated with this document are 5 examples of systematic sensitivity computations which readers can carry out before applying the methods to their own model and simulations. The associated software can be used with a source-code or executable-image version of GEMPACK. The SSA software can be obtained at no cost from the World-Wide Web at address

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

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).

15.2 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.

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 13.6 above was developed in this context.

Note that the range check qualifiers described in section 2.2 are not replacements for the techniques described in Bach and Pearson (1996). 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.

CHAPTER 16

16. Translation between GEMPACK and GAMS Data Files

A facility for transferring data between GAMS and GEMPACK was written and documented by Tom Rutherford (who wrote the GAMS-to-GEMPACK routines) and Ken Pearson (who wrote the GEMPACK-to-GAMS routines) in June 1995. The current version of the GEMPACK-to-GAMS translation is documented in section 16.1 below. Information about the GAMS-to-GEMPACK procedure (which, of course, requires access to a sufficiently recent version of GAMS) can be found in section 16.2 below.

The text of this chapter borrows heavily from the document "Transferring Data Between GAMS and GEMPACK" written by Ken Pearson and Tom Rutherford in June 1995 and circulated to various mailing lists including GAMS-L, GEMPACK-L, MUG-L and GTAP-L.

The authors concentrated on producing a translation which can reliably be read by the other software. (While the form of the data written is not ideal for modellers to read, it is intended to be ideal for accurate transfer between the two software packages.)

16.1 Transferring Data from GEMPACK to GAMS

There are two possible methods, as described below. The "SEEHAR method" is the easier one if your file is suitable.

16.1.1 SEEHAR Method

This method only applies if your **Header Array file has set and element information** in it (see chapter 5 above) and if all sets involved in all coefficients on the file have named elements. In this case, to convert the data to GAMS format, run SEEHAR and select option **GMS**. SEEHAR will then write out the sets and coefficients in GAMS format. [With Release 6.0 of GEMPACK, there are also options **GM1** and **GM2** which produce slightly different GAMS output - for details respond **?GM** to SEEHAR's main menu.]

16.1.2 TABLO Method

If your Header Array file does not have set and element information on it, you can convert it to GAMS format by running TABLO from Release 5.1-003 or later of GEMPACK. To do this, you need to have a TABLO Input file in which the relevant SETs are declared and the relevant COEFFICIENTs declared and have data read into them or assigned via FORMULAs. To write them to a GAMS file you need to declare a suitable logical FILE. Sets are written with a

```
WRITE  (SET)  ... ;
```

statement and COEFFICIENTs are written with a WRITE statement.

Attached is an example file SJTOGAMS.TAB which writes the data for the Stylized Johansen model to a GAMS file. The statements required to transfer data from other models should be clear from this.

Example of writing to a file in GAMS format

```
!-------------------------------------------------------------------!
!               TABLO Input file for writing out                    !
!                the GEMPACK data for Stylized Johansen model       !
!                in GAMS format                                     !
!          (This file is usually called SJTOGAMS.TAB)               !
!-------------------------------------------------------------------!

SET SECT # Sectors # (s1-s2)  ;

SET FAC  # Factors # (labor,capital) ;

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

! Declaration of the GAMS file - this will contain the GAMS
    version of the data !
file (new,gams) output ;

!-------------------------------------------------------------------!
!  Writing the SETs to the GAMS file                                !
!   (Only sets with fixed or run-time elements can be so written)   !
!-------------------------------------------------------------------!
write (set) sect to file output ;
write (set) fac to file output ;

!-------------------------------------------------------------------!
!  Base data and reads                                              !
!   (Base data is as in Table 2.2.1a of GPD-1)                      !
!-------------------------------------------------------------------!

COEFFICIENT   (all,i,SECT)(all,j,SECT) DVCOMIN(i,j)
! Dollar value of inputs of commodity i to industry j ! ;

COEFFICIENT    (all,f,FAC)(all,j,SECT) DVFACIN(f,j)
! Dollar value of inputs of factor f to industry j ! ;

COEFFICIENT (all,i,SECT) DVHOUS(i)
! Dollar value of household use of commodity i ! ;

READ DVCOMIN FROM FILE iodata HEADER "CINP" ;
READ DVFACIN FROM FILE iodata HEADER "FINP" ;
READ DVHOUS  FROM FILE iodata HEADER "HCON" ;

!-------------------------------------------------------------------!
!  Writing the data (coefficients) to the GAMS file                !
!   (Only sets with fixed or run-time elements can be so written)   !
!-------------------------------------------------------------------!
write dvhous to file output ;
write dvcomin to file output ;
write dvfacin to file output ;

!---------end of TABLO Input file---------------------------------!
```

In the TABLO-Input file SJTOGAMS.TAB, the statement

```
file (new,gams) output ;
```

sets up the file with logical name "output" to receive data in GAMS form. The SETs are written with

```
WRITE (SET)
```

statements. The COEFFICIENTs are written with statements of the form

```
WRITE xxx to file output ;
```

The values of the COEFFICIENTs to be written to a GAMS file can have been given via READs (as in SJTOGAMS.TAB above) or via FORMULAs.

A Coefficient can only be written to a GAMS file if all the sets over which its arguments range have element names defined (either specified in the TABLO Input file or read in at run time). For example, the WRITE statement below is not allowed since the set COM does not have element names defined.

```
SET COM SIZE 6 ;
Coefficient (All,c,COM) X1(c) ;
Read X1 from terminal ; file (new,gams) output ;
Write X1 to file output ;  !wrong !
```

Note that WRITEs to GAMS files cannot contain quantifiers. So, for example, the statement

```
WRITE (all,i,SECT) DVCOMIN(i,"s1") to file output ; ! wrong !
```

to write out part of COEFFICIENT DVCOMIN is not allowed.

16.2 Transferring Data from GAMS to GEMPACK

New procedures for transferring data from GAMS to GEMPACK have been written by Tom Rutherford in the second half of 1998. These procedures produce GEMPACK Header Array files with set and element labelling (see chapter 5). [These new procedures rely on a Release 6.0 version of MODHAR being accessible to GAMS.]

Details of these procedures can be found on Tom Rutherford's web site at address

http://robles.Colorado.edu/~tomruth/

or on the GEMPACK web site. Please contact us at the Impact Project if you have difficulty finding details of these procedures.

CHAPTER 17

17. New Command File Statements

In this chapter we show all the Command file statements which are new for GEMPACK Release 6.0 or were new for Release 5.2.

A complete list of all Command file statements allowed can be found in chapter 19.

17.1 New Command File Statements for Release 6.0

```
levels results = YES|no ;    ! see section 2.1


minimum subinterval length = * ;    ! See section 2.2.5
minimum subinterval fails = STOP|continue ;    ! See section 2.2.5


range test updated values = updated|extrapolated|BOTH|no ;   ! See section 2.2.5
range test initial values = YES|no ;   ! See section 2.2.5


statements to shock variable with none exogenous are ok = yes|NO ;
                                              ! See section 8.6


scale equations = yes|NO ;      ! See section 8.5


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 3.3.3


check-on-read coefficients = yes|warn|NO ;      ! See section 5.2 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 8.9
```

17.2 New Command File Statements for Release 5.2

M28 = yes|NO ; ! see section 12.1

SUI = yes|NO ; ! see section 11.7.1
updated formula initial data = <filename> ; ! see section 11.7.1

NWT = yes|NO ; ! see section 12.4

automatic accuracy = yes|NO ; ! see section 12.3
accuracy figures = <d> ; ! see section 12.3
accuracy percent = <pp> ; ! see section 12.3
accuracy criterion = DATA|solution|both ; ! see section 12.3

SSL = all | last | NONE ; ! see section 12.5
SUP = all | last | NONE ; ! see section 12.5

exogenous|endogenous <variable> = zero|nonzero|positive|negative
value on file <file> ; ! see section 13.6
! EXAMPLE.
! exogenous x1 = nonzero value on file DAT1.DAT ;
! endogenous x1 = zero value on file DAT1.DAT ;

xset... ! see chapter 10
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. New TABLO Syntax, Qualifiers

In this chapter we show all the TABLO qualifiers and statements which are new for Release 6.0 or were new for Release 5.2.

A complete list of all Command file statements allowed can be found in chapter 19.

18.1 New TABLO Statement Qualifiers for Release 6.0

`Variable (ORIG_LEVEL = <coefficient-name>)...` New variable qualifier (see section 2.1)

`Coefficient (<operator> <real number >)...` where operator can be **GE, GT, LE, or LT**
New Coefficient or Levels Variable qualifier .(see section 2.2)
EXAMPLE COEFFICIENT(GE 20.0) (all,i,COM) DVHOUS(i) ;

`Write (SET) <setname> to file <logical-file> [ HEADER "<header>"] ;`
! This can now be used to write to a Header Array file or a text file
! not just a GAMS file - see section 4.3

`Write (ALLSETS) to file <hafile> ;` ! Do not specify a Header
! New Write qualifier - see section 4.4

18.2 New TABLO Statement Qualifiers for Release 5.2

`FOR_UPDATES` - new FILE qualifier (see section 11.7)

`WRITE UPDATED VALUE TO …` - FORMULA qualifier (see section 11.7)

`FILE (NEW,GAMS)` - see section 16.1.2

`WRITE (SET)` - see section 16.1.2

18.3 New TABLO Syntax for Release 6.0

Union and Intersection of Sets (section 4.2)

```
SET <setname> [#<labelling-information>#] = <setname1> UNION <setname2> ;
SET <setname> [#<labelling-information>#] = <setname1> INTERSECT <setname2>;
```

18.4 New TABLO Syntax for Release 5.2

Set complements (section 11.1)

```
SET  <new-setname>  =  <bigset> -  <smallset> ;
```

Example. `SET NONMARCOM = COM - MARCOM ;`

Sets depending on data (section 11.2)

```
SET  <new-set>  = (All, <index>, <old-set>: <condition> ) ;
```

Example. SET SPCOM = (all,c,COM: TOTX(c) > TOTY(c)/5) ;

Set Mappings (section 11.3)

```
MAPPING  <set-mapping>  FROM  <set1>  TO  <set2>  ;
```

Example. `MAPPING COMTOAGGCOM from COM to AGGCOM ;`

```
READ (BY_ELEMENTS) <set-mapping> FROM FILE … ;
```
! expects character data !

```
READ <set-mapping> FROM FILE … ;
```
! expects integer data - positions in the set !
! see section 11.3.1 !

Example.

```
READ (BY_ELEMENTS) COMTOAGGCOM from file setinfo header “CTAC” ;
READ COMTOAGGCOM from file textinfo ;
```
! textinfo is a text file !

```
WRITE <set-mapping TO FILE … ;
```
! integer output (numbers) - see section 11.3.5 !

Example. WRITE COMTOAGGACOM to file textoutput ; ! could be to HA file or terminal !

Assertions (section 11.4)

```
ASSERTION  [<qualifiers>] [# message #]
            [<quantifier-list>]<condition> ;
```

Example.

```
ASSERTION # Check no negative values #
      (all,c,COM) DVHOUS(c) >= 0 ;
```

Index expressions (section 11.5)

```
<index-expression-1> <comparison-operator> <index-expression-2>
```

Example.

```
AGGDHOUS(aggc) =
            SUM(c,COM: COMTOAGGCOM(c) EQ aggc, DHOUS(c) ) ;
```

$POS function (section 11.6)

```
$POS(<index>)
$POS(<index>,<set2>)
$POS(<element>,<set>)
```

Writing updated values (section 11.7)

```
FORMULA(INITIAL, WRITE UPDATED VALUES TO FILE <filename>
      HEADER "<HEADER>"
        LONGNAME "<LONGNAME>") <formula> ;
```

Example.

```
  FORMULA (INITIAL,
       WRITE UPDATED VALUE TO FILE upd_prices
                    HEADER "ABCD" LONGNAME "<words>" )
          (all,c,COM) PHOUS(c) = 1 ;
```

CHAPTER 19

19. Summary of Command File Statements

This contains complete and self-contained documentation of all Command file statements for Release 6.0 of GEMPACK.[23]

Command files were introduced in GPD-1 in sections 2.5.5, 2.5.6, 2.8.1 (Figure 2.8.1) and 2.10 (Figure 2.10.4); they can be 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 19.1 below) and SAGEM (see section 19.2 below).

In this chapter, the actual commands are shown in bold; the surrounding text is commentary.

19.1 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.

[23] The material in this chapter is an updated version of the documentation that was in Appendix A of the Release 5.2 edition of GPD-1.

19.1.1 Command File Statements

METHOD AND STEPS

method = johansen|euler|midpoint|GRAGG ; ! gragg is the default
! EXAMPLE. method = gragg ;

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 5.8 of GPD-2 for information about this.)

CHECKING SET AND ELEMENT LABELLING WHEN READING DATA FROM HA FILES

See section 5.2 for details about these.

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 ;

AUTOMATIC ACCURACY

! Automatic accuracy is new for Release 5.2 GEMPACK

automatic accuracy = yes|NO ; ! NO is the default
! use "yes" to specify the accuracy
! see section 12.3 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 2.2.5

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 2.2.5

range test updated values = updated|extrapolated|BOTH|no ;
! 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 afterextrapolation).
! "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 2.2.5

range test initial values = YES|no ; ! 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 2.2.5

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

updated file <logical_name> = <actual_name> ; ! for final updated data

! EXAMPLE. updated file iodata = sjlb.upd ;
! include suffix

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 5.14.1 of GPD-2 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 5.14.1 of GPD-2.
! 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 5.14 in GPD-2 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 7.1 for details.

Equations file = <file_name> ; ! To create and save a new Equations file
! EXAMPLE. Equations file = sj ;
! omit suffix '.eq4'
! In **Release 5.2**. then new BCV file is given same name (but different suffix)
! The following is new for **Release 6.0** GEMPACK
! In general you do not need this statement unless saving an Equations file
! new BCV file is not saved unless statement *BCV file = <file_name> ;* is present
! see section 7.1.4 for details

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

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

BCV file = <file_name> ;
! To specify the name of the (new) BCV (Base Coefficient Values) file to be created.
! In **Release 5.2**, this was only needed when an Equations file is not going to be
! created (perhaps because you using option NSE - see section 5.6.1 of GPD-2
! EXAMPLE. BCV file = sj ;
! omit suffix '.bcv'
! In **Release 6.0**, *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 7.1.4 for details

OTHER FILES

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 7.1 for details

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 7.1 for details

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

! Use 'yes' to get an XAC file
! An XAC file is not created unless you are extrapolating
! from two or three multi-step solutions.
! (In Release 4.2.02 of GEMPACK, these were called 'formatted solution' or 'SLX' files.)

! EXAMPLE. extrap acc file = yes ;

Auxiliary files = <file_name> ;

! (a) GEMSIM (see section 5.9.2 of GPD-2)
! 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 5.9.1 of GPD-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 5.9.1 of GPD-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 Chapter 10 above

xset....

xsubset...

xfile....

xwrite...

xdisplay.....

! 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 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 6.1.2 of GPD-1.

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 ;

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 13.6
! 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 ;

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.)

! 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 6.3.1 of GPD-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 8.6 - This is new for Release 6.0 GEMPACK.

ashock … ; *! Additional shock statement.* For example “ashock xtax = uniform 10 ;”.
See section 8.9 ! This is new for Release 6.0 ! (not allowed with SAGEM)

CUMULATIVELY-RETAINED ROWS

! The use of these commands is discussed in section 6.2.1 of GPD-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)

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 file only = ... ;' is used to direct output to just the Log file (not to the terminal).
! For more details, see section 5.3.3 in GPD-1.

! The following is 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 7.1.2 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 5.6 of GPD-2.

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 donot want these solution results
! see section 2.1 for details.
! This is only available for Release 6.0 GEMPACK.

! 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 sections 5.3-5.9 of GPD-2.)

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 5.7 of GPD-2.)

EAA = yes|no ;
! See section 5.7 of GPD-2 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 ; ! default is YES.
! In Release 5.2 default was NO

! Use 'NSC = yes ;' to delete new BCV
! (Base Coefficient Values) file.
! In **Release 6.0** the default is not to save a BCV file
! see section 7.1.4

NSE = YES|no ; ! default is YES
! In Release 5.2 the default was NO.
! Use 'NSE = yes ;' if you do not want to save an Equations file - see
! section 5.6.1 of GPD-2 for details.
! In **Release 6.0** the default is not to save an equations file
! see section 7.1.4

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.

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

IZ1 = YES|no ; ! default is YES.
! Use 'IZ1 = no ;' to keep coefficients which are zero at step 1.
! This is a change from Release 5.0 when "IZ1 = NO ;" was the default
! see section 5.4.1 of GPD-2.

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.

! The following options were new for Release 5.2 of GEMPACK

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 of GPD-4 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 11.7.1 above for details
! EXAMPLE. sui=yes ;
! updated formula initial value = sjlbfi.upd ;

NWT = yes|NO ; ! default is NO
! Use 'NWT=yes ;' to use the Newton method
! to solve the equations - see section 12.4 above.

SSL= all | last | NONE ; ! default is NONE
! SSL = all ; is used to save files from all separate multi-step
! solutions as well as the extrapolated solution
! SSL= last ; saves just the last of the multistep solutions
! see section 12.5 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 12.5 for details

scale equations = yes|NO ; ! default is NO
! Option new for Release 6.0
! See section 8.5 for details

start with MMNZ|MMNZ1 = <integer value> ; ! Sets initial memory allocation for
! nonzeros in Fortran 90 GEMSIM or TABLO-generated programs.
! See section 3.3.3. New for Release 6.0. Also applicable to SAGEM.

! (See section 5.6.2 of GPD-2 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 5.6.2 of GPD-2.)

DPN = yes|NO ; ! default is NO.
! Use 'DPN = yes ;' to ensure that new pages are
! started in display files only when needed.

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.

19.1.2 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 6.3.3 of GPD-1. 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 7.1.4 for details

19.1.3 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 ;
file iodata = sj.dat ;
updated file iodata = sjlbg.upd ;
extrapolation accuracy file = yes ;
!
! Simulation part
!
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 ;
solut file = sjlbg ;
use Environment file sjxfac ;
shock p_xfac 1 = 10 ;
harwell parameter = 1 ;


verbal description =
Increase labor by 10 per cent.
Standard closure.
Gragg's method - 2,4,6 steps ;
!
! Options (just examples)
!
log file = sjlbg.log ; ! output goes to log file and
                            !  also to terminal
CPU = yes ;           ! report CPU times
!_____________________________________________________________
```

19.2 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 7.1 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 6.1.2 of GPD-1.

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 6.3.1 of GPD-1.

! 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 section 6.2.1of GPD-1.

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 5.3.3 of GPD-1.

simulation = YES|no ; ! default is YES.
! Use 'simulation = no ;' to just specify closure.
! (This corresponds to option NSM - see GPD-1 section 7.6.1.)

start with MMNZ|MMNZ1 = <integer value> ;
! Sets initial memory allocation for
! nonzeros in Fortran 90 SAGEM .
! See section 3.3.3. 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.

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

GENERAL POINTS

[These are just as for Command files for GEMSIM and TABLO-generated programs.]

19.2.1 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 ;
harwell parameter = 0.8 ;
!
! 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 = sag1.log ; ! output to log file and also to terminal
CPU = yes ;           ! report CPU times
!_____________________________________________________________________
```

CHAPTER 20

20. Syntax of GEMPACK Text Data Files

This contains complete and self-contained documentation for GEMPACK text data files.[24]

In preparing arrays of data for GEMPACK text files, a text editor or spreadsheet may be used. Text files can contain several arrays. However, shock files and files used for modifying data with MODHAR via its '**m**' subcommand should only contain one array.

For each array, there must be

- the "how much data" information, followed by
- the actual data values of the array.

20.1 The "How Much Data" Information

This gives information about the array of data values which follow it. It can continue onto second and subsequent lines, can contain comments and must finish with a semicolon ';'. The input of text strings such as 'real' and 'header' is not case sensitive so either 'real' or 'REAL' is correct.

For real or integer data, the expected format is

<sizes> <type> <order> <header> <longname> <coefficient> ;

where

<sizes> is a list of 1 to 7 positive integers, giving the sizes of the array of data values,

<type> is either **real** or **integer**, and

<order> is either **row_order**, **col_order** or **spreadsheet**.

If <type> is omitted, real is assumed.
If <order> is omitted, row_order is assumed.
<header> , <longname> and <coefficient> are optional (see below).

In all cases, real, integer or character, a **semicolon ;** is required at the end of the "how much data" information.

Examples

(1) For a three-dimensional real array of size 2 x 3 x 4 in row order, the "how much data" information is

```
2  3  4  row_order ;
```

(2) For an vector (that is, a 1-dimensional array) containing integers of size 7, the "how much data" information is

```
7  integer  header  "IJK9" ;
```

[24] The material in this chapter is an updated version of the documentation that was in Appendix C of the Release 5.2 edition of GPD-1.

(3) For a real array in column order, with header "ABCD", the "how much data" information (including some comments) could be

```
! This is an array of real numbers in column order
2 3 real col_order
! The header and longname are only read by MODHAR
 header "ABCD" longname "Array description " ;
```

For character string data, the expected format is

<n1> strings length <n2> <header> <longname> ;

where <n1> and <n2> are positive integers specifying respectively the number of strings and the length of each string.

Example

For a list of 3 character strings, each of length 12, the "how much data" information could be

```
3 strings length 12 header "GGGG"
longname "Elements of set G1" ;
```

Header and Longname

In all cases (real, integer or character data), <header> and <longname> are optional. They take the form

HEADER "...."

and

LONGNAME "....."

respectively.

These are only relevant on text files used as input to MODHAR (specifically to the MODHAR commands 'at' and 'aw'). The actual long name (up to 70 characters long) must all be on one line of the file; neither it (nor the actual header) can be split over two lines. If the header and/or long name are not specified on a file which you are reading via MODHAR options 'at' or 'aw', MODHAR will prompt you for them if they are needed.

In all other programs except MODHAR, the header and longname may be present but will be ignored by the program reading the text file.

Coefficient

The final term on the "how much data" information, <coefficient> is only applicable for real data and is also optional. It takes the form

COEFFICIENT <coefficient-name> (<set-name>,<set-name>,....)

or COEFFICIENT <coefficient-name> ("<element-name>" : <set-name>,.....)

where

<coefficient-name> is the name of the coefficient usually associated with this array of data,
<set-name> is the set over which the coefficient is defined,
"<element>" : <setname> indicates that this dimension of the set ranges just over an element and that the element belong to the set following the colon **:**

For example,

coefficient V1BAS(COM, "dom":SOURCE,IND)

or coefficient V1LAB(IND)

This <coefficient> information is only relevant when the text file is being read in MODHAR using the 'at' option. It is used to find which set applies to each dimension of the array and if the set elements have been defined using the 'ds' option in MODHAR , the set and element labelling information can be written to a Header Array as described in section 5.3.4 above. The coefficient term is ignored on text files read into GEMSIM or TABLO-generated programs.

20.1.1 Comments

Any part of an input line following a single exclamation mark ! will be ignored. If the exclamation mark is at the start of the line, the whole line will be ignored. As for terminal input (or input on Stored-input or GEMPACK Command files), each comment finishes at the end of the line it starts on but can be continued by putting ! at the start of the next line.

Comments can be included at the end of any line in the "how much data" information (which may extend over several lines and ends with a semicolon ;).

Comments can also be included anywhere amongst the data values for real or integer data whose order is row_order or col_order. But they cannot be included amongst the data for real or integer data whose order is spreadsheet or with any character data.

```
!         A complete example text file containing 5 arrays
!                 including data values
!
2  3  real  row_order        !  <sizes>  <type>  <order>
HEADER  "abcd"  LONGNAME   !  Longname may be up to 70 characters
!  Write all the longname between double quotes on one line
"The input-output data flows (millions of dollars)"
COEFFICIENT  IOData(COM,IND) ;   ! data follows
4.0  2.0  1.0   ! end of row 1
1.0  6.0  8.0
6   integer  row_order  header  "cdef"  longname
"second array";
1   2    ! One row of data can be split over several lines
3   4    ! Data part of real/integer row/col_order can contain comments
5  6

2,  4,  integer  spreadsheet  header  "fghj"  longname  "third array"  ;
   1,  2,  3,  4,
   5,  6,  7,  8,

2 4  col_order integer header "pqrt"  ;
  1  5
  2  6
  3  7
  4  8

! For a character array, don't leave any blank lines after the semicolon.
! Each of the character strings will be filled with blanks at the
!  right-hand end.

3 STRINGS  LENGTH  5   header "xyza";
abcde
pqr
11223
```

End of example text file

20.2 Data Values in Text Files

20.2.1 Array Sizes

For a 2 x 5 x 7 array, the array sizes are given as three integers at the start of the "how much data" information

```
2 5 7 ! Correct way to give dimensions for 3-dimensional array
```

For a vector (that is, a one-dimensional array), just one size is given since there is no difference between a 1 x 10 array and a 10 x 1 array.

Real arrays can have up to 7 dimensions while integer arrays are restricted to at most 2 dimensions.

20.2.2 Actual Data Values of the Array

The lines of the text file following the "how much data" information contain the actual values for the array.

For real arrays there can be up to 7 dimensions. For a 3-dimensional array of size a x b x c, values are given as if there were c matrices (that is, 2-dimensional arrays) of size a x b.

Here is a 3-dimensional array example, array of size 2x3x4

```
2  3  4  ;
! First the (-,-,1) matrix
1  2  3
4  5  6

! Now the (-,-,2) matrix
7  8  9
10 11 12

! Now the (-,-,3) matrix
13 14 15
16 17 18

! Now the (-,-,4) matrix
19 20 21
22 23 24
```

! In the above example, the (1,3,3) entry is 15

Similarly for a 4-dimensional array of size a x b x c x d, the data is presented as if there are cd matrices each of size a x b (given in the order where the third index ranging over c values varies most rapidly). Each of the 2-dimensional a x b matrices can be given in row or column order depending on what is specified on the "how much data" information.

Note that a matrix X of size m x n has m rows and n columns. The rows are the horizontal lines in the diagram below and the columns are the vertical lines.

```
[   X(1,1)   X(1,2)  ...   X(1,n)  ]   ! row 1
[   X(2,1)   X(2,2)  ...   X(2,n)  ]   ! row 2
      .        .               .
      .        .               .


[   X(m,1)   X(m,2)  ...   X(m,n)  ]   ! row m
```

Row order

- Enter each complete row in the data array in free format, continuing on as many lines of the text file as you like.
- Start each new row on a new line.
- Spaces (or commas) between numbers act as separators.

Column order

- Enter each complete column in the data array in free format, continuing on as many lines of the text file as you like.
- Start each new column on a new line.
- Spaces (or commas) between numbers act as separators.

Spreadsheet

- Spreadsheet order is the same as row order except that usually every data value is followed by a comma as separator.
- There may also be commas after the dimensions in the "how much data" information.
- In preparing the data array on a spreadsheet, enter the data values in the spreadsheet then save the array using the comma-separated values (CSV) format supported by most spreadsheet programs.

One-dimensional Arrays

For a 1-dimensional array, there is no distinction between row and column order; the actual data values can be given on one line or continued on subsequent lines as you like.

20.2.3 Real Numbers

There are various valid ways of writing real numbers:

- as a string of digits with (or without) a decimal point.
  (Whole numbers do not need a decimal point.)
- with positive or negative signs.
- with an exponent followed by an optional sign and an integer.

(For example, 1.23E2 means 1.23x100=123 while 1.23E-2 means

1.23x0.01=0.0123)

Valid examples include

3.2 +1.2 .234567 -15.9E-12 16

Repeated Values in Arrays

Repeated values in arrays can be given in the form

<positive integer>*<real number>

For example, 20*1.35 gives 20 values each of 1.35.

Don't leave spaces on either side of the symbol *.

Don't include brackets around negative real numbers. For negative values the appropriate form is

20*-1.35

When using repeated values, each row or column must still start on a new line in the text file. For example, if you are writing a 10 x 15 array and the last two rows are all zero, it is not correct to use 30*0.0 for the last two rows, you must have two separate rows as in

15*0.0

15*0.0

20.2.4 Character Strings

There should be one character string per line. A character string cannot be broken on a line and continued on the next line. This effectively restricts the maximum length of strings to the maximum length of lines of the text file. Some text editors allow long text lines.

Blank lines after the "how much data" information are important since a blank line counts as a character string filled with blanks.

Comments are not allowed in character data.

20.2.5 Multi-dimensional Real Arrays

Consider a COEFFICIENT C declared as follows in the TABLO Input file.

COEFFICIENT (all,i1,S1)(all,i2,S2)(all,i3,S3)(all,i4,S4) C(i1,i2,i3,i4) ;

The associated array is of size N1 x N2 x N3 x N4 where N1,...,N4 are the sizes of the sets S1,...,S4 respectively. The (4,3,1,2) entry in the array corresponds to the 4th element of S1, the 3rd of S2, the first of S3 and the second of S4.

Suppose that S1,...,S4 have sizes 4,5,2,3 respectively. Then the corresponding data on a GEMPACK text file should be presented as 6 (=2x3) matrices each of size 4 x 5. First are the (1-4,1-5,1,1) entries in 4 rows, then the (1-4,1-5,2,1) entries (since the third index over S3 varies faster than the fourth over S4), then the 4 x 5 matrix corresponding to the (1-4,1-5,1,2) entries, then the (1-4,1-5,2,2) matrix, then the (1-4,1-5,1,3) matrix and finally the matrix corresponding to the (1-4,1-5,2,3) entries of C.

20.2.6 Maximum Length of Lines of a File

There is a limit of 500 characters on the length of lines in the "how much data" information for all three type of files, row_order, col_order and spreadsheet.

For row order or column order files, no line of data (including spaces) on the file can exceed 500 characters (and you may need to be shorter than this on some machines). If you are creating such a file yourself in an editor, it may be best to limit lines to 80 characters (including spaces); each row or column of data can extend over several lines of the file is necessary.

For spreadsheet style arrays, there is no fixed limit. Usually all numbers in one row of data are on one line of the file.

When data in row or column order is read, low-level GEMPACK subroutines process each item of data directly. When data in spreadsheet style is read, a Fortran list-directed read is used.

20.2.7 Differences Between Row Order and Spreadsheet Style Data

In both row)order and spreadsheet data, the entries of the matrices are in row order.

The differences are

(i) in spreadsheet style there is a separator (usually a comma) between data items, and

(ii) in spreadsheet style data, each line of data may be very long. (Ideally, for an N1 x N2 matrix, each line of a spreadsheet style file has N2 data items.)

(iii) no comments are allowed in the actual data for spreadsheet style (see section 20.1.1 above).

When GEMPACK programs (GEMSIM, TABLO-generated programs, SEEHAR or SLTOHT) write spreadsheet style files, they try to put all data entries for one row onto one line of the file (possibly a very long one). For example, in writing a 10 x 112 matrix, there will be 10 lines of data, each containing 112 real numbers.

In contrast, when the same programs write row order data, each line of the file contains at most 5 real numbers. Thus, when writing a 10 x 112 matrix, each of the 10 rows of 112 real numbers is broken into 23 or so lines of the file.

Text editors and text files

It is easy to edit row-order or column-order files in any text editor. However, it may not be possible to edit all spreadsheet style files in your text editor since some lines may be too long for the editor to process. It is best to use spreadsheet style files only for communication between spreadsheet programs and GEMPACK and to use row-order (or column-order) files when you want to edit them in a text editor. (You can always edit spreadsheet files once you have imported them into a spreadsheet program.)

When GEMPACK programs write row-order or column-order data, they put at most 4 or 5 real numbers on each line and separate the numbers with spaces. However, when you are preparing text file for input, you can put more or less numbers per line and can use either spaces or commas (or a mixture) as separators between data items.

Advice for CSV files

If you are saving a CSV file out of a Spreadsheet program such as Excel, put the "how much data" information on the top of the file before you save it to a CSV file. This avoids trouble editing it in a text editor later because the text editor may not be able to cope with the long lines.

Make sure that the type of order is spreadsheet not row_order or col_order.

For example, for a 40x30x4x3 array you must prepare it as rows of 30 numbers. [Each of these rows of data must start on a new line of the file. Any row can be split into several lines.]

The suggested arrangement of this data in the spreadsheet is as 12 matrices each of size 40x30.

Put the separate two dimensional arrays below each other as in

```
40  30  4  3 spreadsheet header "ABCD" longname "Example CSV file" ;

Matrix 1   (40x30)
Matrix 2   (40x30)
Matrix 3   (40x30)
..........
Matrix 12   (40x30)
```

If the elements of the array are indexed by (i1,i2,i3,i4) where
i1=1,...,40, i2=1,...,30, i3=1,...,4, i4=1,...,3

first must come the 40x30 matrix corresponding to i3=1,i4=1
Then must come the one corresponding to i3=2,i4=1
Then must come the one corresponding to i3=3,i4=1
Then must come the one corresponding to i3=4,i4=1
Then must come the one corresponding to i3=1,i4=2

Then must come the one corresponding to i3=2,i4=2
and so on.

[Note that i3 moves faster than i4.]

Once you have prepared this data in the spreadsheet, save it as a CSV file.

20.2.8 Element Name/Number Labels

When GEMPACK programs (GEMSIM, TABLO-generated programs, SEEHAR or SLTOHT) write row-order or column-order real or integer data, they put element names (or numbers if names are not available) amongst the data to indicate the position in the array of each number. These labels are written as comments (following the style indicated in section 20.1.1 above).

Note that these element name or number labels are not required when you prepare a text file to be read. Indeed, even if included, they are ignored since they are treated as comments when such a file is read; in particular, programs reading files with element names as comments do not check that the names are correct.

Below we give an example of a 4 x 7 array to show these labels. Note that the row element labels are at the end of the first line of data for each row.

Example of Element Names

```
!
! Values of TO_HAT(NX_COMM,REG) - an array of size 4x7
!------------------------------
!
! (Entries are written one "row" at a time, maximum of 4 per line.)

4  7  real row_order ;

! The matrix TO_HAT(%1,%2) with %1 in NX_COMM, %2 in REG. +++++++

!%2= Australasia  USCN           HiYEAsia       NewNICs
  5.960465E-06   0.000000E+00   1.192093E-05  0.000000E+00 !%1=land
!%2= China        EC12           ROW
  0.000000E+00   0.000000E+00   0.000000E+00

!%2= Australasia  USCN           HiYEAsia       NewNICs
  0.000000E+00   5.960465E-06   0.000000E+00  1.192093E-05 !%1=labor
!%2= China        EC12           ROW
  1.192093E-05   0.000000E+00   1.192093E-05

!%2= Australasia  USCN           HiYEAsia       NewNICs
  0.000000E+00   5.960465E-06   1.192093E-05  0.000000E+00 !%1=capital
!%2= China        EC12           ROW
  1.192093E-05   5.960465E-06   0.000000E+00

!%2= Australasia  USCN           HiYEAsia       NewNICs
 -2.31080       -14.2302       -22.3250      -1.59832     !%1=crops
!%2= China        EC12           ROW
 -6.96980       -30.2941      -1.192093E-05
```

20.3 Converting Release 4.2.02 Files for Use with Release 5 or Later

This section tells users with text files prepared for use with Release 4.2.02 of GEMPACK how to convert these files for use with Release 5 or later. (Readers not familiar with Release 4.2.02 should ignore this section.)

We apologise to users of Release 4.2.02 for the fact that the format of text files required for Release 5 differs from that for Release 4.2.02. We believe we made a mistake in requiring column order in Release 4.2.02 (we still find it difficult to work with) and we believe that the current files will prove much easier to work with. The essential difference is that, with Release 4.2.02 we insisted on column order, whereas with Release 5 or later we allow you to use either row order or column order.

Although text file formats differ between these releases, it is most important to realise that Header Array files created using Release 4.2.02 or earlier releases (including those prepared using text files) do not need any change for use with Release 5. However, if you wish to change a Release 4.2.02 text file and use it with Release 5 (or later) GEMPACK software to create a new Header Array file, you will need to change the text file. This is very easy to do, as explained below.

Because Release 4.2.02 of GEMPACK required what is now referred to as column-ordered files, the simplest way to modify Release 4.2.02 text files is just to change the "how much data" parts of them to

- include 'col_order',
- omit the "number of dimensions" integer at the start, and
- add a semi-colon at the end of this.

The actual data part should not need any changes. (Just check that each new "column" of data starts on a new line.)

Example

Consider the Release 4.2.02 text data file containing one real array of size 2 x 3 x 4. The "how much data" information on it will be

```
3 2 3 4
```

Simply change this to

```
2 3 4 col_order ;
```

(That is, omit '3' at the start and add 'col_order ;' at the end.)

You might also like to add comment lines to make these files more self-documenting.

CHAPTER 21

21. Choosing Sets of Variables Interactively[25]

When running GEMPIE or SLTOHT, you may need to choose sets of variables interactively (or via a Stored-input file). This has been introduced in section 6.2.2 of GPD-1. Here we list complete details of the procedures for making these choices.

Sometimes your choice will be simple and easily expressed. For example, you may wish to choose all variables or none, or all "macro" variables (that is, variables with just one component), or just a few variables. Procedures for doing these have been described in section 6.2.3 in GPD-1. On the other hand, you may need to describe a fairly complicated set - perhaps many variables and only some of the components of some of them. GEMPACK provides you with two ways of making such a choice.

1. The first is the "active" way in which you give **lists** of the variables (and their components) you want to be in the subset.
2. The second is the "passive" way in which you let the software **prompt** you about all the available variables in turn and you say, for each one, which components of it you wish to be in the subset.

Of these two, the Lists option is the default and the one we recommend to you. The Lists option, which has been described briefly in section 6.2.4 in GPD-1, is described in detail in section 21.2 below while the "responding to prompts" method is described in section 21.3 below.

You will recognise the choice situations we are describing in this section because you will be presented with a menu looking something like that shown below.

Example of the Menu Presented for Choosing a Subset

```
    CHOICE OF WHICH cumulatively-retained endogenous VARIABLES YOU WANT
                  TO BE printed.
Make ONE of the following choices:
   L     LISTS of variables, all or some of whose
           components are to be printed.
   a     ALL components of ALL cumulatively-retained endogenous
           variables to be printed.
   m     All cumulatively-retained endogenous MACRO
            variables to be printed.
   1     All components of ONE cumulatively-retained endogenous
           variable to be printed.
   f     All components of a FEW cumulatively-retained endogenous
            variables to be printed.
   s     SOME components of SOME cumulatively-retained endogenous
           variables to be printed.
   w     WHICH are the cumulatively-retained endogenous variables.
Enter your choice now. ('L' is the default.)
```

(In the menu above, which comes from GEMPIE, the words "cumulatively-retained endogenous" and “printed” will be replaced, more generally, by words describing the big set and the subset being chosen.)

[25] The material in this chapter was in section D.2 of Appendix D of the Release 5.2 edition of GPD-1. [Note that the material in section D.1 of the Release 5.2 edition of GPD-1 is no longer included in the Release 6.0 documentation since we think you should never need to choose a closure interactively, but rather always choose closures using a Command file.]

(The menu above is the one you will see if you are using full prompts. If you have chosen brief prompts, as described in section 5.4 of GPD-1, the menu will be compressed considerably.)

Choices a,n,m,1,f,w have been described in section 6.2.3 of GPD-1, choice L (Lists) is described in section 21.2 and choice s (SOME) is in section 21.3 below.

21.1 ***Specifying Components of One Variable***

In either of the 'L' (Lists - Active) or 's' (Some - Passive) methods, you may wish to specify just some of the components of one variable.

Under 'L' this can be done by specifying components via set/element arguments, as in, for example,

a p_XF("labor",SECT)

Alternatively, you can use subcommand 's' under the 'L' options and specify component numbers as indicated below.

The procedure for specifying component numbers directly is described below.

First you will be asked **how many** components you wish to choose. Respond with an integer.

Then you will be asked for the **actual components**. These you must specify by number, and you can abbreviate several consecutive components using, for example, 6-10 to indicate components 6,7,8,9,10. Your response can extend over several lines if you wish and can contain several such groups of consecutive components or single component numbers, separated by commas or spaces, as you prefer. For example, the following two lines specify 85 of the 120 components of a particular variable.

1-6, 8 12 16 21-40,
50 - 70 86-120

See section 6.1.3 of GPD-1 for the order of the components of variables with two or more arguments.

21.2 ***The Lists Option - Choosing a Set Actively***

Once you have chosen 'L' (Lists) from the main menu, you can use several commands (as described below) to specify the set. This choice is done in a command-driven mode in which (until you indicate that you have finished the choice) you see the following prompt.

> Next choice (or ? for help)

The commands available are listed below, together with a brief description of each one. (If you ask for help, you get essentially the table below except that the MEANING column will give a more explicit description that reflects the current context.)

Subcommands Available When Choosing via the Lists Option

COMMAND	MEANING
a <list>	Choose ALL components of the listed variables.
s <list>	Choose SOME components of the listed variables.
m	Choose all available MACRO variables.
w	Which are available components of variables to choose from?
c	Which components of which variables have I already chosen?
q	Finish.
Q	Quit.(Give up this attempt to choose a set of variables.)
?	Help.
? <list>	Detailed help for the specified commands.

These commands are described in more detail in the following pages. An example should help to make it clear.

EXAMPLE

Suppose that you are choosing the cumulatively-retained endogenous variables to be printed (as in the menu shown early in this chapter) and suppose that the model has the following cumulatively-retained endogenous variables:

- x1 4 components, all cumulatively-retained endogenous
- x2 6 components, numbers 1,3,5 of which are cumulatively-retained endogenous
- y1 10 components, all cumulatively-retained endogenous
- y2 1 component, cumulatively-retained endogenous
- z1 3 components, all cumulatively-retained endogenous
- z2 3 components, all cumulatively-retained endogenous

The 2 commands

a x1 x2 y2
s y1 z1

followed by, responding (as described in section 21.1 above) to prompts about first 'y1' and then 'z1',

4 (for 'y1')
1-3 9

2 (for 'z1')
1, 2

will say that all exogenous components of x1, x2 and y2 are to be printed and that just components numbered 1-3 and 9 of 'y1' and components 1 and 2 of 'z1' are to be printed. Then the command

f (to finish)

completes the choice of the set of variables to be printed.

In the command

a <list>

you can choose whole classes of variables by indicating their arguments and the sets over which they range. For example

a (COM) (COM,IND)

chooses all (available) components of

- all variables in the model having one argument ranging over the set COM [this is what **(COM)** means], and
- all variables in the model having two arguments ranging respectively over the sets COM and IND [this is what **(COM,IND)** means].

(Of course, in such cases, the sets and arguments are as contained on the TABLO Input file for the model.) The set name or names must be enclosed in brackets '(' and ')' in this type of command.

Note that, as you issue more commands, you keep adding to the subset.

SUBCOMMANDS AND DETAILED EXPLANATION

(a) To choose ALL (available) components of a list of variables

Command: **a <list>** (for 'Choosing ALL components')

<list> is a list (separated by spaces) of names of the variables which are to have all components selected.

<list> can also contains names of variables followed by sets or elements as arguments, for example p_XF("labor,SECT), to specify just the relevant components. In this case there must be no space between the end of the variable name and the "(" starting the arguments.

<list> can also contain classes of variables specified via their arguments such as (COM) or (COM,IND).

(b) To choose SOME (available) components of a list of variables

Command: **s <list>** (for 'Choosing SOME components')

<list> is a list (separated by spaces) of names of the variables which are to have some components selected.

Then, for each variable in the list (in the order you have listed them), you will be prompted for:

The number of components to be chosen.

The ACTUAL components. (Follow the method in section 21.1 above.)

(Note that if you enter an invalid response at any time with the 's' command, NO components of that variable will be chosen.)

(c) To choose all available MACRO variables

Command: **m** (for 'Choosing MACRO variables')

(d) To list which components of variables are available for selection

Command: **w** (for 'Which')

(e) To list which components of variables have been selected so far

Command: **c** (for 'Chosen')

(f) To finish selecting a subset of variables

Command: **f** (for 'Finish')

(g) To Quit an attempt to choose a set of variables

Command: **q** (for 'Quit')

(Note that if you quit you will lose any changes you have made, and will be able to start choosing the set of variables again.)

(h) To get help with Choosing commands

Commands: **?** (for 'A listing of available commands')

? <list> (for 'Detailed help for command(s)')

<list> is the list of commands you require help with.

NOTE. If, in options (a) or (b) above, one of the names in your list is not in the model or if it has no components in the set from which you are choosing, this name will be ignored (a message will be given). Note that the other (legal) variables listed will still have all or some of their components chosen if you are running interactively; however if you are running in batch (or from a Stored-input file), the program will stop once this invalid input is encountered.

21.3 The Some Option - Responding to Prompts

If you respond 's' (Some) in the main menu, you will be presented with the names of the variables (those with at least one component in the big set) one at a time. For each one, you are first asked to respond

a to select ALL (available) components to be in the subset,
n to select NO components to be in the subset, or
s to select SOME (available) components to be in the subset.

If you respond 's', you specify the components you want by following the method described in section 21.1 above.

Note that the variables are presented in the order in which they are declared on the TABLO Input file for the model in question. This information should help you anticipate the order of the prompts if you are creating a Stored-input file. (But remember that variables with no components in the big set are omitted in this prompting.)

If you do create a Stored-input file, we suggest you comment it by putting the names of the variables as comments (precede them with a single exclamation mark) after the a/s/n responses; this makes it easier to modify and, when you run the program, you can check that the prompts and your answers coincide.

22. 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.

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. 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.

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.

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.

23. GEMPACK DOCUMENTS[26]

Harrison, W.J. and K.R. Pearson (1998), *An Introduction to GEMPACK*, GEMPACK Document No. 1 **[GPD-1],** Monash University, Clayton, Fourth edition, October 1998.

Harrison, W.J. and K.R. Pearson (1994), *User's Guide to TABLO, GEMSIM and TABLO-generated Programs*, GEMPACK Document No. 2 **[GPD-2],** Monash University, Clayton, Second edition, April 1994.

Harrison, W.J. and K.R. Pearson (1993), *How to Create and Modify GEMPACK Header Array Files Using the Program MODHAR*, GEMPACK Document No. 3 **[GPD-3],** Monash University, Clayton, Third edition, April 1993.

Harrison, W.J. and K.R. Pearson (1998), *Release 6.0 of GEMPACK - New Features and Changes From Release 5.1 and 5.2*, GEMPACK Document No. 4 **[GPD-4],** Monash University, Clayton, Second edition, October 1998.

Harrison, W.J. and K.R. Pearson (1998), *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, Eighth edition, October 1998.

Harrison, W.J. and K.R. Pearson (1998), *Installing and Using the Executable-Image Version of GEMPACK on DOS/Windows PCs*, GEMPACK Document No. 7 **[GPD-7],** Monash University, Clayton, Fifth edition, October 1998.

Harrison, W.J. and K.R. Pearson (1998), *Getting Started with GEMPACK: Hands-on Examples*, GEMPACK Document No. 8 **[GPD-8],** Monash University, Clayton, First edition, October 1998.

[26] 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".

INDEX

$POS, 11-78
Abbreviation
  Command file syntax, 19-127
Absorption, 1-6
Acceptable Range
  Coefficient, 2-15
ACCUM, 6-46
Accuracy
  User-specified, 12-83
Additional shocks, 8-59
Aggregating Data
  Mapping, 11-71
Aggregating simulation results, 11-71
Array sizes
  Text file, 20-140
ashock statements, 8-59
ASSERTION, 8-56, 11-76
Automatic accuracy, 19-115
Auxiliary files
  Check, 13-91
Auxiliary Statement, Table files, 19-118
BCV file, 7-53, 8-57
  Saving, 7-53, 8-57
  Use not supported in next release ?, 7-53, 8-57
  Using, 7-53, 8-57
Center for Global Trade Analysis, 9-64
Character strings
  Text file, 20-142
Check
  ASSERTION, 11-76
Check-on-read
  defaults, 5-33
Check-on-read statements, 5-32
Choice of variables
  Sets, 21-149
Closure
  Command file syntax, 19-120
  Seeing, 13-91
  Sets, 10-65
  When Processed, 13-90
-cmf option, 1-5, 7-54
Coefficient
  Acceptable Range, 2-15
  Argument - Index Expression, 11-74
  How much data information, 5-35
  Integer, Display, 13-90
Column order
  Text file, 20-141
Command file, 19-113
  Example, 19-129
  SAGEM example, 19-135
  Statements, 17-110
Command file syntax
  Abbreviation, 19-127
  Automatic accuracy, 19-115
  Case independent, 19-127
  Closure, 19-120
  Files, 19-117
  SAGEM, 19-130
  Shocks, 19-122
  TG Programs and GEMSIM, 19-114
  Verbal description, 19-121
Command files
  Extra statement, 10-65
  Minimum subinterval statements, 2-17
  Range test, 2-17
Command line options
  -cmf, 1-5, 7-54
  -log, 1-5, 7-54
  -lon, 1-5, 7-54
  only under DOS, 1-5, 7-54
  -sti, 1-5, 7-54
Comments, 10-65
  Command file syntax, 19-127
  Mapping files, 1-4
  Text file, 20-139
Comparing indices, 11-77
Compatability
  Release 5.1, 5-36
Complement
  Set, 11-69
Component list
  Shocks, 19-122
Components of variables
  Specifying, 21-148
Composition of Mappings, 11-74
Compressed form
  RWHAR, 13-94
CSV file
  GEMPACK text file, 20-143
Data
  Extrapolation Accuracy Summary, 12-83
Data file
  Command file syntax, 19-116
Defaults
  Command file, 19-127
Define sets
  MODHAR, 5-35
DEVIA, 6-46
Display
  Command file, 10-65
  Integer coefficient, 13-90
Displays, 13-98
Dynamic models, 6-46, 8-59
Element names
  SLTOHT, 6-39
  Text file, 20-144
Empty Sets, 1-4, 11-70
Environment
  SEENV, 13-91
Equations
  Scaling, 8-56
  Solving them, 12-81
Equations file
  Command file syntax, 19-117
  Saving, 8-57
  Using with BCV file, 7-53, 8-57
Error
  Extra statement, 10-66
Executable-image licence limits, 1-7
Extra data file
  Command file syntax, 19-116

Extra statements, 10-66
Extrapolation Accuracy Summary, 12-83
  Data, 12-81
  Levels variables, 12-82
FILE
  Default names, 13-89
  FOR_UPDATES, 11-79
  GAMS, 16-105
  Parameter, 13-90
Forecasting models, 6-46, 8-59
FORMULA(INITIAL)
  Save updated values, 11-79
  WRITE(UPDATED), 11-78
Fortran 90
  Memory management, 3-19
GAMS
  Translation, 16-105
GEMPACK for Windows, 9-61
GEMPACK licences
  Executable-image licence limits, 1-7
GEMPIE
  Levels results, 2-13
GEMSIM
  Command file syntax, 19-113
  Increasing parameters, 14-100
  Set and element information, 5-31
GEMSIM and TG Programs
  Command file example, 19-129
Getting started with GEMPACK, 9-62
GPSTRT, 14-101
Graphs
  Using SLTOHT, 6-45
GTAP, 9-64
Harwell Laboratories, 12-81
Homogeneity
  Example, 13-95
  SUMEQ, 13-95
How much data information
  Coefficient, 5-35
Index Expression, 11-74
  Comparison, 11-77
Index expression comparison, 11-71
Inequality, 15-103
Integer coefficient
  Display, 13-90
Intermediate file
  Command file syntax, 19-116
Intersection
  Set, 4-24
Intertemporal models
  Semantics, 4-25
Keyword
  Extra statement, 10-65
Labelling
  Set and element, 5-29
Large-simulations licence, 1-7
  When required, 1-7
Levels equations
  Newton's method, 12-85
Levels models, 2-12
Levels results
  GEMPIE, 2-13
  SLTOHT, 2-13
  ViewSOL, 2-13
Levels values
  Reporting, 2-11
  SLTOHT, 6-41
Levels variable
  Acceptable range, 2-15
Levels variables
  Pre and post-simulation, 2-11
Licence file, 14-100
Licences
  Executable-image licence limits, 1-7
Line length
  Text file, 20-142
Log file
  Error, 13-89
-log option, 1-5, 7-54
-lon option, 1-5, 7-54
Longnames
  Write statements, 8-55
Low memory code
  SCS, LIR, LRP, 14-101
LSTHDS, 14-102
MA28
  Speed, 8-55
MA48, 12-81
Macro variables, 21-147, 21-150
Mapping, 11-70
Maximum size
  Set, 4-23
MEGABARE model, 6-46, 8-59
Memory
  Insufficient, 3-19
Memory Allocation
  Fortran 90, 3-19
MKHAR
  Set end element information, 5-31
MMNZ, 14-100
MMNZ and MMNZ1, 3-21
MMNZ1, 14-100
MODHAR
  Defining sets, 5-35
  Set and element information, 5-30, 5-35
MONASH model, 6-45, 6-46, 8-59
  SLTOHT option SHK, 6-45
Newton variable
  $del_newton, 12-85
Newton's Method, 12-85
Options
  GEMPIE - NLV, 2-13
  GEMSIM and TG Programs - SSL, SUP, 12-87
  GEMSIM and TG Programs - SUI, 11-79
  GEMSIM and TG Programs M28, 12-81
  RWHAR - C51, 5-36, 13-94
  RWHAR - CPR, 13-94
  SAGEM - Command file, 19-134
  SEEHAR - CPL, 13-89
  SEEHAR - CPW, 13-89, 13-98
  SEEHAR - DEC, 13-89
  SEEHAR - FNP, 13-89
  SEEHAR - SSE, 5-29
  SLTOHT - NEL, 6-40
  SLTOHT - SHK, 6-45
  SLTOHT - SHL, 2-13
  SLTOHT - SSS, 6-39
  TABLO - CDM, 14-100

TABLO - NTX, 8-58
TABLO - NWT, 12-86
TABLO - W77, 3-20, 14-100
Options GEMPIE - SNA, 2-13
Options List
GEMSIM and TGPrograms, 19-124
Out of memory, 3-19
Parameter files, 13-90
Parameter values
Increasing, 14-100
Parameters MMNZ and MMNZ1, 14-100
Pivots
Reuse, 13-91
Position Number
$POS, 11-78
Post-simulation coefficients, 2-11
Preliminary pass
Set sizes, 3-20
Pre-simulation coefficients, 2-11
Prompts
Some option, 21-151
Quotas, 15-103
RE type (data on HA file), 5-30
Real arrays
Multi-dimensional, 20-142
Real Numbers
Text file, 20-141
Repeated values, 19-128
Text file, 20-141
Reuse of Pivots, 13-91
Row order
Text file, 20-141
Row sums
Homogeneity, 13-95
RunGEM, 9-61, 9-63
RunGTAP, 9-64
RWHAR
Set end element information, 5-31
SAGEM
Command file example, 19-135
Command file syntax, 19-130
Save Updated values
SUI, 11-79
Scaling
Equations, 8-56
SEEHAR, 13-98
Set and element labels, 5-29, 5-30
SEENV, 13-91
Sensitivity Analysis, 15-103
Set
Command file, 10-65
Complement, 11-69
Data dependent, 11-70
Empty, 1-4
Intertemporal, 4-25
Mapping, 11-70
Maximum size, 4-23
Position Number, 11-78
Union, Intersection, 4-24
Writing elements, 4-24
Set and element information, 5-29
Adding, 5-34
Adding in MODHAR, 5-35
Removing, 5-36, 13-94
Set element name
Reading, 10-66
Set Mapping
Aggregating data, 11-71
Aggregating simulation results, 11-71
Set/element information
checking when reading data, 5-31
Set/element labelling
Types RE/RL/2R, 5-30
Sets
Empty, 11-70
Sets of variables
Choosing, 21-149
Setting MMNZ and MMNZ1, 3-21
Set-up file, 14-101
Several Subintervals, 12-84
Shocks
Command file syntax, 19-122
Component list, 19-122
No shock statements, 14-99
Repeated values, 19-128
Select from file, 19-122
Some exogenous, 14-99
When Processed, 13-90
Shocks for Command files
SLTOHT, 6-45
Short spreadsheet output
SLTOHT, 6-39
SLTOHT
Levels results, 2-13
Spreadsheet Mapping files, 6-39
SLTOHT Mapping files
Rules, 6-43
Software
SSA, 15-103
Solution
Extrapolation Accuracy Summary, 12-83
Solution file
Saving multi-step, 12-87
Solution method
Command file syntax, 19-114
Verbal description, 13-90
Solving
Levels equations, 12-85
Sparse Matrices
MA48, 12-81
Spreadsheet
Preparing tables, 6-39
Text file, 20-141
Spreadsheet Mapping file, 13-92
Spreadsheet output, 13-98
Spreadsheet separator, 13-98
SSA, 15-103
Steps
Command file syntax, 19-114
Verbal description, 13-90
-sti option, 1-5, 7-54
Subintervals, 12-83, 12-84
Command file syntax, 19-114
Submatrix data, 14-101
Subtotal
Command file syntax, 19-133
SUMEQ
Example, 13-95

Homogeneity, 13-95
Tables
3-dimensional, 6-41
Several variables, 6-42
SLTOHT - SSE, 6-40
TABLO Code
Low memory, 10-67
TABLO-generated programs
Command file example, 19-129
Command file syntax, 19-113
Fortran 90 or 77, 3-20
Increasing parameters, 14-100
Set and element information, 5-31
TABLO-like statements, 10-66
Terminal data
Automatic accuracy, 8-56
Test Coefficient
Range, 2-16
Text file
Data values, 20-140
How much data information, 5-35
Line length, 20-142
Spreadsheet CSV, 20-143
Union
Set, 4-24
Updated data
Saving multi-step, 12-87
Set and element information, 5-31
Updated data file
Parameter, 13-90
Updated values
FORMULA(INITIAL), 11-78
user-specified accuracy, 2-15, 12-83
Variable
Argument - Index Expression, 11-74
VARIABLE(ORIG_LEVEL=..), 2-11
Variables
Macro. *See* Macro variables
Specifying components, 21-148
Verbal description, 13-90
Command file, 19-121
ViewHAR, 9-61
ViewSOL
Levels results, 2-13
W77 option
TABLO, 3-20
WinGEM, 1-8, 9-61, 9-62
Write
Command file, 10-65
Updated values, 11-78
WRITE(ALLSETS), 4-25
WRITE(SET), 4-24
XAC-retained variables, 12-84
XDISPLAY, 10-65
XFILE, 10-65
XSET, 10-65
Complement, 11-70
XSUBSET, 10-65
XWRITE, 10-65