GEMPACK USER DOCUMENTATION

Release 7.0

GPD-4 Useful GEMPACK Programs

Useful GEMPACK Programs

GEMPACK Document No. GPD-4

W. J. Harrison

K. R. Pearson

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

First edition
October 2000

© Copyright 2000. The Impact Project and KPSOFT.

First edition
October 2000

ISSN 1030-2514

ISBN 0-7326-15054

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

Abstract

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

This document describes how to use various utility GEMPACK programs including: MODHAR, GEMPIE, SLTOHT, SEEHAR, SEENV, SUMEQ, RWHAR, MKHAR. These programs process files used in GEMPACK programs such as Header Array files and Solution files. These files cannot be created, read or modified in a text editor because they are binary files.

This document also contains a brief description of the various Windows programs including WinGEM, RunGEM, ViewHAR, ViewSOL, TABmate and AnalyseGE which are available on PCs.

Document Attributes

Name	: Useful GEMPACK Programs
Audience	: CGE Modellers
Identifier	: GPD-4

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

In the Release 6.0 documentation, GPD-4 was used for the GEMPACK document entitled "Release 6.0 of GEMPACK : New Features and Changes from Release 5.1 and 5.2" (1998).

Table of Contents

1. THE OTHER GEMPACK PROGRAMS 1-1

1.1 Licence Files 1-2

2. GEMPACK WINDOWS PROGRAMS 2-3

2.1 WinGEM - the Windows Interface to GEMPACK 2-3

2.2 ViewHAR for Looking at or Modifying Data on a Header Array File 2-4

2.3 ViewSOL for Looking at Simulation Results on a Solution file 2-5

2.4 TABmate for Working on TABLO Input files or Command files 2-6

2.5 RunGEM for Model Users 2-7
2.5.1 TABmate and RunGEM for Model Developers? 2-8
2.5.2 Preparing a Model for Use with RunGEM 2-8
2.5.3 Systematic Sensitivity Analysis via RunGEM 2-8

2.6 AnalyseGE – Assisting in the Analysis of Simulation Results 2-9

2.7 RunDynam for Recursive Dynamic Models 2-10

2.8 Getting Started with GEMPACK using WinGEM or RunGEM 2-10

2.9 New Features with Release 7.0 2-10

3. MODHAR: CREATING AND MODIFYING HEADER ARRAY FILES 3-11

3.1 Header Array Files 3-11
3.1.1 Array Type 3-11

3.2 Programs Used with Header Array files 3-12

3.3 An Overview of the Use of MODHAR 3-13
3.3.1 Creating a New Header Array File 3-13

3.4 Modifying an Existing Header Array File 3-14
3.4.1 Adding Arrays from Other Files using MODHAR 3-14

3.5 USING MODHAR Commands 3-15

3.6 Commands for Creating a New Header Array File 3-16
3.6.1 Commands for Modifying an Existing Header Array File 3-17
3.6.2 Commands for Adding Arrays to the New File 3-18

3.7 Operations on Headers or Long Names 3-21

3.8 Commands for Complicated Modification or Addition 3-23
3.8.1 MODHAR Sub-commands for *mw* or *aw* 3-23
3.8.2 Examples 3-24

3.8.3 Writing Arrays to the New Header Array File 3-26
3.8.4 Option ADD when Modifying Data in MODHAR 3-27

3.9 Finishing Up 3-27
3.9.1 Exit Commands 3-27
3.9.2 History of the New File 3-28

3.10 Complete Example of a MODHAR Run 3-29

3.11 Text Files 3-30

4. VIEWING OR COMPARING HEADER ARRAY FILES 4-31

4.1 SEEHAR: Prepare a Print File or CSV Spreadsheet File 4-31
4.1.1 SEEHAR - Options 4-33
4.1.2 Displays and Labelled Output from SEEHAR into Spreadsheets 4-33
4.1.3 GAMS Output from SEEHAR, TABLO-generated Programs and GEMSIM 4-33
4.1.4 Converting Header Array Files to Data Bases - Option SQL 4-34

4.2 CMPHAR: Comparing Data on Header Array Files 4-35
4.2.1 CMPHAR Report 4-35
4.2.2 Significant Differences 4-35
4.2.3 Comparing Solution Files 4-35

4.3 SUMHAR: Summarizing a Header Array File 4-36

4.4 CMBHAR: Combining similar Header Array Files 4-36

5. SET AND ELEMENT LABELLING ON HEADER ARRAY FILES 5-37

5.1 Effect of Set and Element Information on GEMPACK Programs 5-38
5.1.1 SLTOHT 5-38
5.1.2 SEEHAR 5-38
5.1.3 MODHAR 5-38
5.1.4 TABLO-generated Programs and GEMSIM 5-39
5.1.5 RWHAR and MKHAR 5-39

5.2 Checking Set and Element Information when Reading Data 5-39

5.3 Adding Set and Element Information to an Existing HAR File 5-39
5.3.1 Method 1 - Extra Write Statements 5-39
5.3.2 Method 2 - Using a Special TABLO Input file 5-40
5.3.3 Method 3 - Using Updated files 5-40
5.3.4 Method 4 - Using MODHAR 5-40
5.3.5 Method 5 - Using ViewHAR 5-42

5.4 Compatibility with Earlier Releases of GEMPACK 5-42

5.5 Technical Point 5-42

6. SYNTAX OF GEMPACK TEXT DATA FILES 6-43

6.1 The "How Much Data" Information 6-43
6.1.1 Real or Integer Data 6-43
6.1.2 Examples of "How Much Data" Information 6-44
6.1.3 Character String Data 6-44

6.1.4 Header and Longname 6-44
6.1.5 Coefficient 6-45
6.1.6 Comments 6-45
6.1.7 Complete Example of GEMPACK Text File 6-45

6.2 Data Values in Text Files 6-47
6.2.1 Array Sizes 6-47
6.2.2 Actual Data Values of the Array 6-47
6.2.3 Row Order 6-48
6.2.4 Column Order 6-48
6.2.5 Spreadsheet Order 6-48
6.2.6 Real Numbers 6-48
6.2.7 Repeated Values in Arrays 6-49
6.2.8 Character Strings 6-49
6.2.9 Multi-dimensional Real Arrays 6-49
6.2.10 Maximum Length of Lines of a File 6-50
6.2.11 Differences Between Row Order and Spreadsheet Style Data 6-50
6.2.12 Text Editors and Text Files 6-50
6.2.13 Advice for CSV Files 6-51
6.2.14 Element Name/Number Labels 6-51

6.3 Slight Incompatibility Between Release 5.2 and 6.0 Text Files 6-52

7. GEMPIE: PRINTING SIMULATION VARIABLE RESULTS 7-53

7.1 Using WinGEM to Run GEMPIE 7-53

7.2 Using GEMPIE Interactively 7-53
7.2.1 Solution Files Produced by GEMSIM or TABLO-generated Programs 7-54
7.2.2 Solution Files Produced by SAGEM 7-55

7.3 Choosing Sets of Variables Interactively When Running GEMPIE 7-56

7.4 General Points About Output from GEMPIE 7-56

7.5 GEMPIE Options 7-57

8. SLTOHT: PROCESSING SIMULATION RESULTS 8-59

8.1 Solution File to Header Array/Text/Spreadsheet file 8-59

8.2 SLTOHT Examples 8-59
8.2.1 Totals Solution to Header Array File 8-59
8.2.2 Totals Solution to CSV File for Spreadsheet (SSE) 8-60
8.2.3 Cumulative + Levels Solution to Header Array File 8-61
8.2.4 Subtotals + Cumulative Solution to Header Array File 8-62

8.3 Mapping Files 8-63
8.3.1 Example of a Spreadsheet Mapping file (Option SSS) 8-63
8.3.2 Example of Matrix Variables for Spreadsheets (Option SSE) 8-64

8.4 Running SLTOHT Interactively 8-64
8.4.1 Choosing Solutions for Output 8-64
8.4.2 Shocks and Undefined Solution Values 8-65

8.5 Levels Results – Options NLV and SHL 8-65

8.6 Header Array file Output 8-66
8.6.1 Header Mapping file 8-66
8.6.2 Alternative Header Array File Output - Variable Arguments Ignored (VAI) 8-67

8.7 GEMPACK Text File Options (SIR,SIC), including Spreadsheet (SS) 8-68

8.8 Shock CMF Output - Option SHK 8-70

8.9 Other SLTOHT Options 8-70

9. SLTOHT FOR SPREADSHEETS 9-73

9.1 Spreadsheet Mapping Files 9-73
9.1.1 Syntax Rules for Spreadsheet Mapping Files 9-75

9.2 Options Spreadsheet (SS) and Short Spreadsheet (SSS) 9-76
9.2.1 Component Numbers rather than Element Names (SS or SSS) 9-76

9.3 Output of Tables Using Option SSE 9-77
9.3.1 Output of Levels Results Using Option SSE 9-78
9.3.2 Three and Higher Dimensional Arrays of Results 9-78

9.4 Tables Containing Results for Several Variables (SS or SSS output) 9-79

9.5 Tables Suitable for Producing Graphs 9-82

10. ACCUM AND DEVIA : ACCUMULATION AND DIFFERENCES 10-83

10.1 Use with Dynamic Forecasting Models 10-83
10.1.1 Using ACCUM to Prepare a Spreadsheet Table for the Forecast 10-84

10.2 ACCUM 10-84
10.2.1 Columns Showing Totals and Averages 10-84
10.2.2 Year-On-Year Results or Accumulated Results (Option ACC) 10-85
10.2.3 Accumulated Indexes (Option ACI) 10-86

10.3 Using DEVIA to Prepare a Spreadsheet Table for the Differences 10-86
10.3.1 Making Sure DEVIA Knows Change/Percent-Change Variables (Option SOL) 10-86
10.3.2 Year-On-Year Differences (Option NAC) 10-87

11. TRANSFERRING MODELS BETWEEN MACHINES 11-89

11.1 Transferring the Whole Model 11-90

11.2 RWSOL, MKSOL: Transferring Solution Files 11-91

11.3 RWHAR, MKHAR: Transferring Data Files 11-91
11.3.1 RWHAR - Options CPR, C51 11-91

11.4 Transferring Johansen Simulation Capability 11-92

11.5 Comparing Results from Different Machines 11-92

12. SEENV: TO SEE THE CLOSURE ON AN ENVIRONMENT FILE 12-93

12.1 Command File Output 12-93

12.2 Spreadsheet Output 12-94

12.3 Shock Statements for the Bottom of a CMF File 12-94

13. SUMEQ: INFORMATION FROM EQUATIONS FILES 13-95
13.1.1 Map of an Equations File Produced via SUMEQ 13-96

13.2 SUMEQ and the Homogeneity of Models 13-97
13.2.1 Why Using SUMEQ is Better than Carrying out a Simulation 13-97
13.2.2 Homogeneity Example with Miniature ORANI 13-98
13.2.3 SUMEQ for Homogeneity - Summary 13-100

14. TEXTBI : EXTRACTING TEXT FILES FROM BINARY FILES 14-101

15. CONVERTING BINARY FILES: LF90/F77L3 AND LF95 15-103

15.1 Converting Between Lahey and Fujitsu Binary Files 15-103

15.2 GEMPACK Programs on PCs Can Handle Both Type of Files 15-103
15.2.1 Names and Location of These Converted Files 15-104

16. TRANSLATION FROM GEMPACK TO 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. CHOOSING SETS OF VARIABLES INTERACTIVELY 17-109

17.1 Simple Choices 17-110

17.2 Specifying Components of One Variable 17-110

17.3 The Lists Option - Choosing a Set Actively 17-111
17.3.1 Lists Option - The Subcommands in More Detail 17-113

17.4 The Some Option - Responding to Prompts 17-114

18. REFERENCES 18-115

19. GEMPACK DOCUMENTS 19-116

20. INDEX 20-117

CHAPTER 1

1. The Other GEMPACK Programs

This document is a guide to other useful GEMPACK programs, that is, those apart from TABLO, GEMSIM, SAGEM and TABLO-generated programs. TABLO is documented in GPD-2 and the simulation programs, GEMSIM, SAGEM and TABLO-generated programs, are documented in GPD-3.

These other programs are used to process various special GEMPACK files which are binary files and which therefore cannot be read in a text editor or printed out.

All these utility programs are supplied with the Source-code version of GEMPACK, though executable images of some are not created as part of the standard installation on Windows PCs. If you are working with a Source-code version of GEMPACK on a Windows PC, and the executable image of the program is not present in your GEMPACK directory, consult the GEMPACK document GPD-6 to see how to make a main program.

Many of these programs are supplied as part of the Executable-image version of GEMPACK. Some are supplied as part of the Demonstration Version of GEMPACK.

Some of these useful GEMPACK programs are available from the GEMPACK Website

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

Chapter 2 contains brief introductions to the various **Windows programs** (which are only available for PCs running Windows 95, 98, NT or 2000) which are usually supplied with GEMPACK:

WinGEM	Windows interface to GEMPACK
ViewHAR	Windows program for looking at data in a Header Array file
ViewSOL	Windows program for looking at Solution files
RunGEM	Windows program for automating simulations with models
TABmate	Windows text editor for developing TABLO Input files
AnalyseGE	Windows program assisting modellers to analyse their results
RunDynam	Windows interface for recursive dynamic models.

The remaining chapters and programs are grouped in this document according to the type of file that is being discussed or that the programs are processing.

Data Files (Header Array files and Text Data files) - see chapters 3 to 6

Header Array files see chapters 3 to 5:

MODHAR	for building or modifying data files
ViewHAR	for examining data on Header Array files or for modifying such data files
SEEHAR	seeing the Header Array file
SUMHAR	summarising a Header Array file
CMPHAR	comparing Header Array files
CMBHAR	combining similar Header Array files

Text data files (chapter 6)

Simulation Results (Solution files) - see chapters 7 to 10:

GEMPIE	for printing simulation results
SLTOHT	for post-solution processing of simulation results
ACCUM, DEVIA	for accumulating results and finding deviations

Software for transferring models between different machines (that is, different operating systems) - see chapter 11:

MKHAR, RWHAR	transferring Header Array files
MKEQ, RWEQ	transferring Equations files
RWSOL, MKSOL	transferring Solution files
CMPSOL, COMPEQ	comparing files

Environment and Equations files - see chapters 12 and 13:

SEENV	for determining the closure from Environment or Solution files
SUMEQ	for information about the Equations file

Software for extracting a text file from certain GEMPACK binary files (for example, for extracting the TABLO Input file stored on an Auxiliary Table file or the Command file stored on a Solution file):

TEXTBI	(see chapter 14)

Software on Windows PCs for converting between the binary files (most importantly Header Array files) produced by F77L3 and LF90 on the one hand and LF95 on the other hand:

CONLF (see chapter 15)

Procedures for converting between GEMPACK and GAMS data files are described in chapter 16.

Chapter 17 is really an appendix that describes the methods for choosing sets of variables interactively. This method can be used in various programs including GEMPIE and SLTOHT.

1.1 Licence Files

You do not to require a licence file for the GEMPACK utility programs[1]. Only TABLO, GEMSIM and SAGEM require access to a GEMPACK licence, and also large TABLO-generated programs in some circumstances (see section 1.2.5 of GPD-1).

Some of the newer more powerful options in the Windows GEMPACK programs ViewSOL, ViewHAR, TABmate are only available if you have a Release 6.0 (or later) licence. Systematic sensitivity analysis in RunGEM also requires a Release 6.0 (or later) licence.

AnalyseGE requires a Release 7.0 (or later) licence except for very small Solution files. You need to run simulations using a Release 7.0 version of GEMSIM or TABLO-generated program in order to create the Solution Coefficients file that is used by AnalyseGE. This type of file, which is new for Release 7.0, is described in section 8.4 of GPD-3.

[1] New licence files are required for Release 7.0 of GEMPACK.

CHAPTER 2

2. GEMPACK Windows Programs

This chapter is a brief introduction to the GEMPACK Windows programs. Each of the Windows programs has its own **on-line Help** which you can consult for more details.

You can install all these Windows programs from your GEMPACK CD following instructions in GEMPACK document GPD-6 if you have a Source-code version of GEMPACK, or in GEMPACK document GPD-7 if you have the Executable-Image version.

The Windows programs ViewHAR, ViewSOL and TABmate were all written by Mark Horridge, a colleague at the Centre of Policy Studies. All three have been considerably improved with new features to make data and solution handling easier.

The Windows programs distributed with Release 7.0 (or later) of GEMPACK are all designed to run under the operating systems Windows 95, 98, NT or 2000. [They do not run under the older Windows 3.1.]

2.1 WinGEM - the Windows Interface to GEMPACK

WinGEM, the Windows interface to GEMPACK[2], 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. 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, implementation using TABLO -> carrying out a simulation -> running ViewSOL to print 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.]

To use WinGEM, you need to have installed on your computer

- the WinGEM program and its associated text editor GemEdit,
- some version of GEMPACK (Source code, Executable Image or Demonstration), and
- the Windows programs ViewHAR, ViewSOL and TABmate.

For Release 7.0 and later, WinGEM loads TABmate when the TABLO Input file being checked has syntax or semantic errors; you work in TABmate to correct the errors and then return to WinGEM.

For Release 7.0, WinGEM can handle long file names. However, if you are using a Source code version of GEMPACK with the Fortran compiler F77L3, you will run into trouble because the

[2] WinGEM was introduced with Release 5.2 of GEMPACK. WinGEM can only be run successfully if your GEMPACK programs are from Release 5.2 (or later).

compiler can only handle the original (8.3) format for both file and directory names. See section 4.9 of GPD-1 for details of allowed file names.

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

ViewHAR is a Windows program which allows you to examine the data on a Header Array file in a Windows fashion. You can also use ViewHAR 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 (2 or 1 dimensions) 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), this will be available through ViewHAR.

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, or into a wordprocessor such as Word.

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

ViewHAR has on-line help which includes the following topics:

Looking at files
Modifying files
Guide to the simplified ViewHAR Menu
Guide to the full ViewHAR Menu

ViewHAR will work with the Charter, a companion program which turns numbers from ViewHAR and ViewSOL into graphs. See Charting in the on-line help.

You need a GEMPACK licence to use some of the more advanced features of ViewHAR -- such as file editing or charting.

You can open a GEMPACK text file directly in ViewHAR.

There are two other ways to look at real matrices - see Special Views in the on-line help:

Sparse Sorted View: which sorts each row in ascending order, and
Levels with Shares, which puts 2 numbers (share and level) in each cell.

Note that ViewHAR can also read and write GEMPACK text data files (see chapter 6), as well as Header Array files. Given a Header Array file containing Set and Element labelling and also Coefficient Names, ViewHAR can also write a simple TABLO Input file see section 1.4.2 of GPD-2 for an example of this feature.

You can aggregate data in ViewHAR. ViewHAR makes it easy to set up (or read in) the set mappings for the aggregation you want. You can view the aggregated data and/or save it to a new Header Array file. See the chapter “Aggregating or Remapping Data” in the on-line ViewHAR Help file.

ViewHAR can read both types of Header Array files (Lahey or Fujitsu – see chapter 15).

2.3 ViewSOL for Looking at Simulation Results on a Solution file

ViewSOL is a Windows program for looking at results on GEMPACK Solution files (that is, ViewSOL is an alternative to GEMPIE). In fact ViewSOL can be used with more than one Solution file so you can view side by side different Solution files and compare results.

ViewSOL can show subtotals results (see chapter 11 of GPD-3) and levels results (see section 8.3 of GPD-3). You can ask ViewSOL to ignore these via the ***File | Options*** menu.

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.

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, or into a wordprocessor such as Word. ViewSOL can also be used to produce charts of variables easily.

Occasionally it may seem that ViewSOL is not showing up-to-date results. If this seems to be the case, select ***File | Options*** and make sure that "Keep SOL and SLI files" is not checked.[3] Then open the Solution file again.

Some users have been puzzled because ViewSOL does not always show the results for all variables on the Solution file. This is controlled by "Max size variables to show" under the ***File | Options*** menu. Set this value to 0 if you want all variables shown (even ones with large numbers of components).

ViewSOL has on-line help that includes the following topics.

Overview
The Contents Window
The Data Window

Arranging vectors by size or name
Filtering for particular set elements
Viewing Several Solutions at Once
Current Solution

Mouse Actions on Variable Names
Re-Ordering Variables
Hiding Variables

Charts

The Format Menu Item
Format Files

Copying to ClipBoard
Number of Decimal Places
Printing
Altering font size

[3] When ViewSOL opens a Solution file, it runs the GEMPACK program SLTOHT (see chapter 8) to convert the results to a Header Array file with suffix .SOL. When "Keep SOL and SLI files" is checked, ViewSOL does not recreate the SOL file if, on the basis of various time and date checks, the SOL file seems to be up-to-date. This speeds up ViewSOL but occasionally means that ViewSOL may not show up-to-date results when "Keep SOL and SLI files" is checked.

2.4 TABmate for Working on TABLO Input files or Command files

TABmate is a Windows program which is used to edit text files, especially TABLO Input files and Command files. TABmate has its own on-line Help file.

In the GEMPACK modelling system, text files suffixed *.TAB* are used to specify economic models. TABmate is designed to be used for modifying and debugging these TAB files. It is aimed at the intermediate or advanced GEMPACK user, and is particularly useful for large or complex TAB files. The special features of TABmate are:

- Colours, bolding and italics are used to highlight the syntactical structure of the TAB file. This makes it easier to understand the TAB file, and to spot errors within it. For example, keywords are rendered in bold; a mis-spelt keyword will not show bold and so attracts attention. Comments appear in blue -- so it easy to see which sections of the TAB file have been commented out.
- Without leaving TABmate, you can quickly locate any syntax or semantic errors in a TAB file. TABmate uses the GEMPACK program TABLO to scan the TAB file. It marks any errors and displays TABLO error messages.
- After running TABLO, you can click on any variable, coefficient or set and press the Gloss button to display a list of every statement in the file which mentions that variable, coefficient or set. For a coefficient, say, the first few of these statements will usually furnish a definition. The remainder show how the coefficient is used. Line numbers accompany each statement; you can click on these to go straight to that location in the TAB file.
- Tables of Coefficients, Variables and Equations can be generated which may be used to document your model.

For Release 7.0 and later, WinGEM loads TABmate when the TABLO Input file being checked has syntax or semantic errors; you work in TABmate to correct the errors and then return to WinGEM.

Recent versions of TABmate make it easy for you to write a TABLO-generated program (via the Code button) and to compile and link it.

We think that TABmate plus RunGEM provide a good environment for model developers – an alternative to WinGEM. See section for more details.

2.5 RunGEM for Model Users

RunGEM is a Windows program[4] 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 you 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 you can

- choose the model you 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 you 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.
- carry out systematic sensitivity of their results with respect to variations in parameter or shock values (see section 2.5.3 below).

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, you 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, you can use the mouse to

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

Users of RunGEM need have no prior knowledge of GEMPACK or the syntax of Command files. You 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 - you 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.

---

[4] RunGEM was new for Release 6.0 of GEMPACK.

Whenever RunGEM 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.

A hands-on introduction to RunGEM can be found in chapter 5 of GPD-8.

RunGEM 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.

2.5.1 TABmate and RunGEM for Model Developers?

We think that TABmate plus RunGEM provide a good working environment for model developers. Some model developers may prefer this to the environment provided by WinGEM. Use TABmate to debug your TAB file, then to write code and compile and link it. Use RunGEM for simulations.

2.5.2 Preparing a Model for Use with RunGEM

Modellers with a Source-code or Executable-image version of Release 6.0 (or later) 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.)
- the Model Information file produced when TABLO is run. [This file, which has suffix .MIN, is always produced when the Release 6.0 (or later) 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.

2.5.3 Systematic Sensitivity Analysis via RunGEM

The version of RunGEM distributed with Release 6.0-001 or later of GEMPACK makes it very easy for users to carry out systematic sensitivity analysis with any models implemented and solved using GEMPACK.[5] You can compute how sensitive your simulation results are to changes in the underlying parameters (for example, Armington parameters) of your model, or to changes in the shocks.

To obtain documentation and help for this facility, click on

Tools | Help on Systematic Sensitivity Analysis (SSA)

in RunGEM's main menu.

[5] The first version of RunGEM incorporating systematic sensitivity analysis was version 1.2 (April 1999).

2.6 AnalyseGE – Assisting in the Analysis of Simulation Results

AnalyseGE is a Windows program designed to assist modellers in the analysis of their simulation results.[6]

When you carry out an application with a general equilibrium model, you are faced with the task of explaining the results. You need to identify and quantify the main mechanisms of the model which are producing the results. To do this, details of the equations of the model, the base data, consequences of that data (totals etc) and the simulation results (percentage changes etc) are all needed. Often these are found in different places on the computer (different files etc), and a time-consuming part of the task is moving between the different sources of information.

AnalyseGE is a software tool which is aimed at assisting modellers to move quickly between these different information sources. It can be used to analyse the results of any simulation carried out using Release 7.0 (or later) of GEMPACK. The AnalyseGE interface gives you "point and click" access to the equations of the model, the data and consequences, and to the simulation results[7]. In particular a modeller can click on any equation and ask the software to group the terms into different natural parts, and give the numerical values of each term.

AnalyseGE can only be used to analyse simulation results held on Solution files and Solution Coefficients files produced with Release 7.0 (or later) of GEMPACK.

There are three separate Windows open when AnalyseGE is running:

1. the AnalyseGE window
2. the TABmate window
3. the ViewHAR window

Usually you operate in the TABmate window where the TABLO Input file of your model is displayed. When you *point and click* in this special TABmate window, the results appear in the ViewHAR window. You can bring any of these three Windows to the front.

A Hands-on introduction to AnalyseGE in the context of a well-known simulation with GTAP can be found in Pearson *et al* (2000). A hands-on introduction to AnalyseGE in the context of models usually distributed with GEMPACK can be found in chapter 6 GPD-8.

[6] AnalyseGE was introduced with Release 7.0.

[7] If you have condensed your model, you can not see variables in AnalyseGE which have been substituted out. However you can see the values of backsolved variables so you may decide to alter the usual condensation of your model to change from substituting large variables to backsolving.

2.7 RunDynam for Recursive Dynamic Models

RunDynam is a Windows interface which is designed to automate the use of recursive dynamic models. There are also model-specific versions of RunDynam, including RunMONASH and RunGTEM.

RunDynam is not distributed freely in the same way as the other Windows programs described in this chapter. If you are interested in using RunDynam, or having a version customised for a particular model, please consult the GEMPACK developers.

RunDynam allows to you to construct a base case (which may be a forecast) and policy deviations from the base case with a model which has been implemented using GEMPACK. The model is solved on a year-to-year basis (that is, recursively) over a number of years, starting from initial data. For each subsequent year, the starting data is the updated data produced by the previous simulation.

You first solve the base case, then carry out the policy deviation. With some models it is customary to rerun the base case before carrying out the policy deviation.

You can choose any group of input data files for the model; these are the starting points for your base case. You specify closures and shocks on text files, using the syntax required in GEMPACK Command files. You can choose names for the output files from the base case and policy runs. You can choose among several methods to solve the model. You can view the results of the base case or policy deviation on the screen, or export them to other programs. You can view or copy either the initial model data base or any of the updated data files produced during the base case or policy deviation.

See also section 10.1 about recursive dynamic models. The MONASH model is described in detail in Dixon and Rimmer (2000).

2.8 Getting Started with GEMPACK using WinGEM or RunGEM

Chapter 2 of the GEMPACK document GPD-8 *Getting Started with GEMPACK: Hands-on Examples* contains a step-by-step introduction to GEMPACK via WinGEM. Chapter 5 of GPD-8 contains instructions for using RunGEM on two example models.

2.9 New Features with Release 7.0

These Windows programs described in the following sections have been made more user-friendly in a number of small ways : for example,

- ViewHAR: more support for working with sets and mappings, including a built-in data aggregation facility.
- ViewSOL: now indicates if a variable is of ordinary change type (as opposed to percentage change). You can view the verbal description associated with each solution. The *Keep SOL and SLI files* option (which enables ViewSOL to show results more quickly if it has opened the same solution before) is now more reliable.
- TABmate: The Cross-Reference or Gloss feature now has two modes: it either lists every statement in the file which mentions a particular symbol; or it lists definitions of every variable, coefficient or set mentioned in the current statement. There is special colouring for CMF files. There are tools to assist in backing up or deleting files.
- WinGEM and other programs Long filenames and directories allowed unless using Fortran F77L3
- RunGEM Systematic Sensitivity Analysis available easily

CHAPTER 3

3. MODHAR: Creating and Modifying Header Array Files

This chapter contains an introduction to Header Array files (section 3.1) and to the programs which can be used to create or modify Header Array files (section 3.2). The GEMPACK program usually used to modify data on a Header Array file is MODHAR. The rest of the chapter (sections 3.3 to the end) is a complete documentation of the program MODHAR.

Chapter 4 documents the programs which can be used to look at or compare data on Header Array files.

3.1 Header Array Files

Data for economic models (for example, input-output tables or parameters such as elasticities) are often held within GEMPACK on files called Header Array files. Header Array files contain one or more arrays each containing data values. An individual array of data on a Header Array file is accessed by supplying the unique 4-character identifier (or Header) for that array of values.

The data values held on an individual array can be either all real numbers, all integer numbers or all character strings. Depending on the type of data that is to be stored, the number of dimensions allowed varies.

The dimension limits for Header Arrays are :

- For real numbers - up to and including 7 dimensions
- For integer numbers - up to and including 2 dimensions
- For character strings - only one dimensional arrays are allowed, each string being of the same length

Headers for arrays on any one file must be unique since the header acts as a label or primary key to identify the associated array.

Once written, an array contains not just the data for the array itself but also self-describing data, including the type of data values, dimensions of the array and a descriptive "long name" of up to 70 characters.

Header Array files have the advantage that you can access any array just by referring to the header which uniquely identifies the array in the file. There is no need to consider here the format of the arrays or any other details since they are all taken care of automatically by the software.

Headers consist of four characters which are usually letters (A to Z, a to z) or digits (0 to 9). Different arrays must have different headers. The case (upper or lower) of a header is not significant. (For example, you cannot have one array with header 'ABCD' and another on the same file with header 'AbCd'.). Headers starting with letters 'XX' are reserved for internal program use so an error is returned if you choose a header starting with 'XX'.

3.1.1 Array Type

Each array on a Header Array file has an associated type. The type of each array is shown when SEEHAR or ViewHAR lists the contents of a Header Array file.

Arrays of integers have type **2I**, arrays of character strings have type **1C**. Arrays of reals can have any of the types **2R**, **RL** or **RE**. Type **RE** indicates that the array has set and element labelling (see chapter 5).

3.2 Programs Used with Header Array files

If you need to view data on Header Array files, the Windows program **ViewHAR** (see section 2.2) can be used on a PC. You can use the GEMPACK program **SEEHAR** (on all computers) as described in section 4.1.

This chapter deals mainly with the topic of creating or modifying data on Header Array files. The program MODHAR (which is documented in sections 3.3 to the end of this chapter) is the one usually used for these purposes.

If you need to modify data values in one or more arrays on a Header Array file, there are various alternative methods to be considered as alternatives to using MODHAR. While these methods may be convenient for minor editing of values, if this is a task you are likely to have to repeat one or more times as the input data changes, it is best to use MODHAR or the TABLO method and Stored-input files rather than a Windows programs where you have no record of the mouse clicks necessary to repeat the task later.

1. Spreadsheet method

Input the arrays to a spreadsheet. If the original arrays are on a Header Array file, the program SEEHAR is available to convert them to a form suitable for input to a spreadsheet (see section 4.1 for more details.)

Edit the arrays in the spreadsheet then output them as text arrays using the comma-separated values (CSV) format (see chapter 6). They can then easily be converted to a Header Array file using MODHAR or ViewHAR.

2. TABLO Method

Write a TABLO Input file to read values of the arrays from Header Array files (or text files) into arrays defined as COEFFICIENTs. Use FORMULAs to manipulate the data in various ways then write the arrays to a Header Array file. (See section 3.4.3 of GPD-1 for an example.)

3. ViewHAR and Spreadsheets

By use of the Advanced Editing Menu in ViewHAR, you can export values from a two-dimensional array or two dimensional section of an array to a spreadsheet. After editing, the array values can be pasted back into the Header Array in ViewHAR.

A recent development is that you can use ViewHAR to modify one or more entries of arrays on a Header Array file by right-clicking on the value.

The rest of this chapter (sections 3.3 to the end) documents MODHAR.

3.3 An Overview of the Use of MODHAR

Data on Header Array files cannot be edited (that is, modified) using normal text editors. The data on a Header Array file is encoded in binary form to keep the size of the file small. MODHAR is therefore provided to enable you to create a new Header Array file that is based on, but differs from, an existing Header Array file. MODHAR can also be used to create a new Header Array file.[8]

The usual method of running MODHAR is **interactively** whether you are working in WinGEM or at the Command prompt. However if you are carrying out a complicated edit using MODHAR, use the SIF option to create a Stored-input file containing your responses. If you need to rerun MODHAR to carry out the same task, you can then run MODHAR using the STI option. See section 4.3 of GPD-1 for details of STI and SIF.

3.3.1 Creating a New Header Array File

When MODHAR is being used to create a new file, the basic method of operation is to read data values from some source into memory and then write them as header arrays to the new file.

Data values can be read in from text files or the terminal as shown in Figure 1. Arrays of data can also be copied from existing Header Array files to the new file. In this mode of operation the data values in the arrays are not usually modified, merely copied unchanged from the source to the new file.

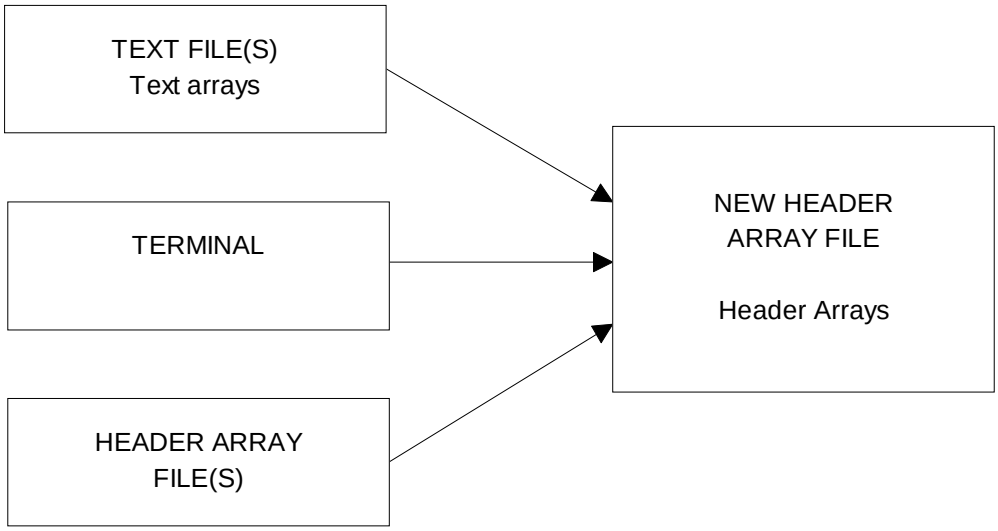


Figure 1: Creating a New Header Array File

[8] Prior to Release 7.0, the documentation of MODHAR was contained in GEMPACK document GPD-3 "How to Create and Modify GEMPACK Header Array Files Using the Program MODHAR" (April, 1993).

3.4 Modifying an Existing Header Array File

One very common need for GEMPACK users is to modify data on an existing Header Array file. The arrays on the file cannot be edited directly; instead they are copied from the old existing file to a work array in memory. Here the data values in the array can be changed and finally can be written as Header Arrays to the new file. As shown in Figure 2, extra arrays can also be added from extra Header Array files (not just the old file) and from extra text files.

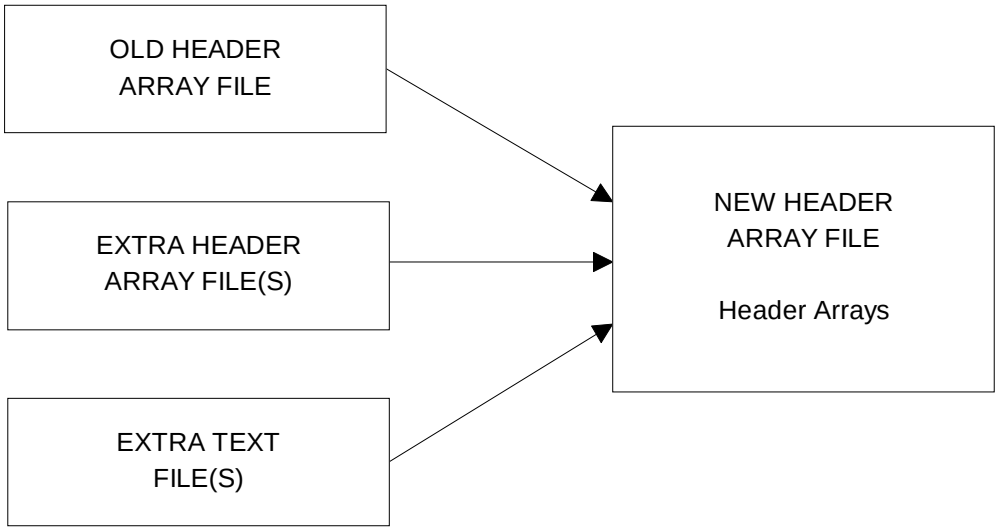


Figure 2: Modifying an Existing Header Array File

Arrays on the old file which are not modified can be conveniently copied directly to the new Header Array file as a block.

Note that unlike some other packages (such as word processors or spreadsheets), MODHAR does not actually modify the existing Header Array file. Rather, MODHAR leaves it unchanged and creates a new Header Array file containing the changes you make. (For this reason, the new file should be given a different name from that of the existing one.)

3.4.1 Adding Arrays from Other Files using MODHAR

MODHAR includes simplified procedures (via the commands **at** and **ah**) for adding arrays from text files, extra header array files and from spreadsheets.

Using the format of GEMPACK text data files described in chapter 6 you can

- add several arrays from the same text file.
- include on text files the header and long name associated with each array, and read them in with the array. This makes it a quick and simple procedure to read in all the arrays on a text file at once. Alternatively you can skip over arrays on the text file if you don't want them included on the new Header Array file.
- specify the order of data on text files as row-order, column-order or spreadsheet. The aim of this is to make it easy to transfer arrays of data between spreadsheets and Header Array files since one of the most convenient ways of editing data is on a spreadsheet.

3.5 USING MODHAR Commands

MODHAR is a command-driven program and is easy to use. The menus of possible actions clearly state the tasks that you can currently perform.

MODHAR commences with the usual sort of GEMPACK options screen (as described in section 4.3 of GPD-1). Then you are asked if the new file to be created is to be based on an old (that is, an existing) file.

Old or New File

(i) If you are modifying an old file, respond '**y**', and you will then be asked for the name of the file you wish to modify.

(ii) Otherwise (that is, you are creating a new file), respond '**n**'.

Note that any references in the rest of this document to the "old file" are only relevant in case (i) above.

The MODHAR Command Menu offers the following commands:

MODHAR Commands

ah	add one or more arrays from Header Array file
ds	define sets for adding arrays from a text file
at	add one or more arrays from text file
da	delete one or more arrays from old file
ra	restore one or more deleted arrays from old file
mw	modify data from the original file and write it on the new file
aw	add new data and write it on the new file
ex	exit, saving the new Header Array file
q	quit, without saving the new Header Array file
lh	list the headers, their status and size of associated arrays
oh	show one header, its status and size of associated array
ch	change a header name
eln	examine a long name
cln	change the long name associated with a header
?	help

MODHAR commands fall into four groups

- Commands for simply adding arrays to the new file **ah**, **at, ds**.
- Commands for carrying out operations on headers or long names **lh, oh, ch, eln, cln, da, ra.**
- Commands for complicated modification or addition of data on arrays from the old file, terminal or other files **mw, aw.**
- Exit commands **ex or q**.

3.6 Commands for Creating a New Header Array File

When MODHAR is being used to create a new Header Array file, it is usually just a matter of transferring data from elsewhere (text files, other Header Array files or input from the terminal) to the new Header Array file. This process is illustrated in Figure 3 where you can see that this can be accomplished using the MODHAR commands at, ah and aw. Once all the desired data has been added, you use the exit command ex to complete the task.

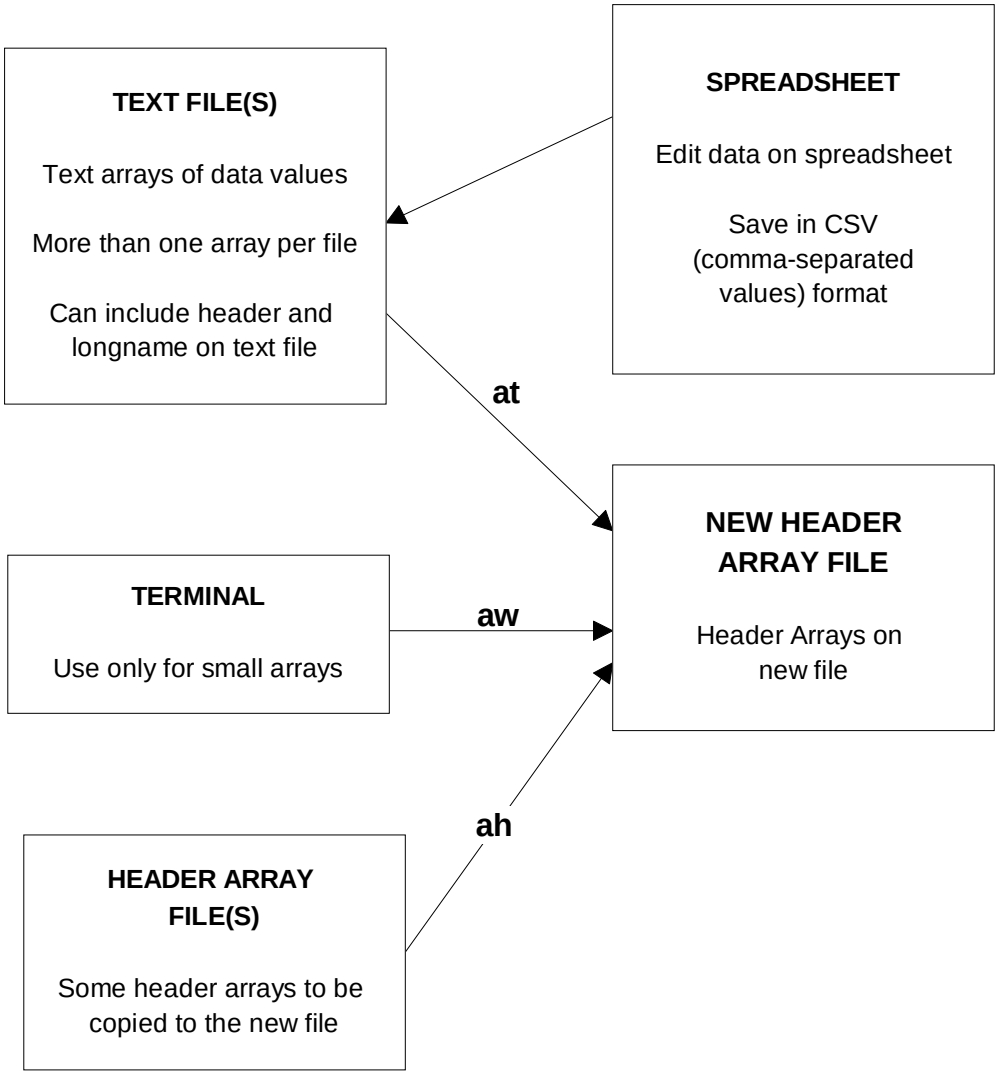


Figure 3: Commands for Creating a New Header Array File

3.6.1 Commands for Modifying an Existing Header Array File

Here you may need to use the full range of MODHAR commands, as illustrated in Figure 4.

Although you are "modifying an existing HeaderArray file", the old file is not modified or changed. We just mean that you start with an existing Header Array file. A new file is created containing the arrays that you ask to be transferred from the existing file to the new file.

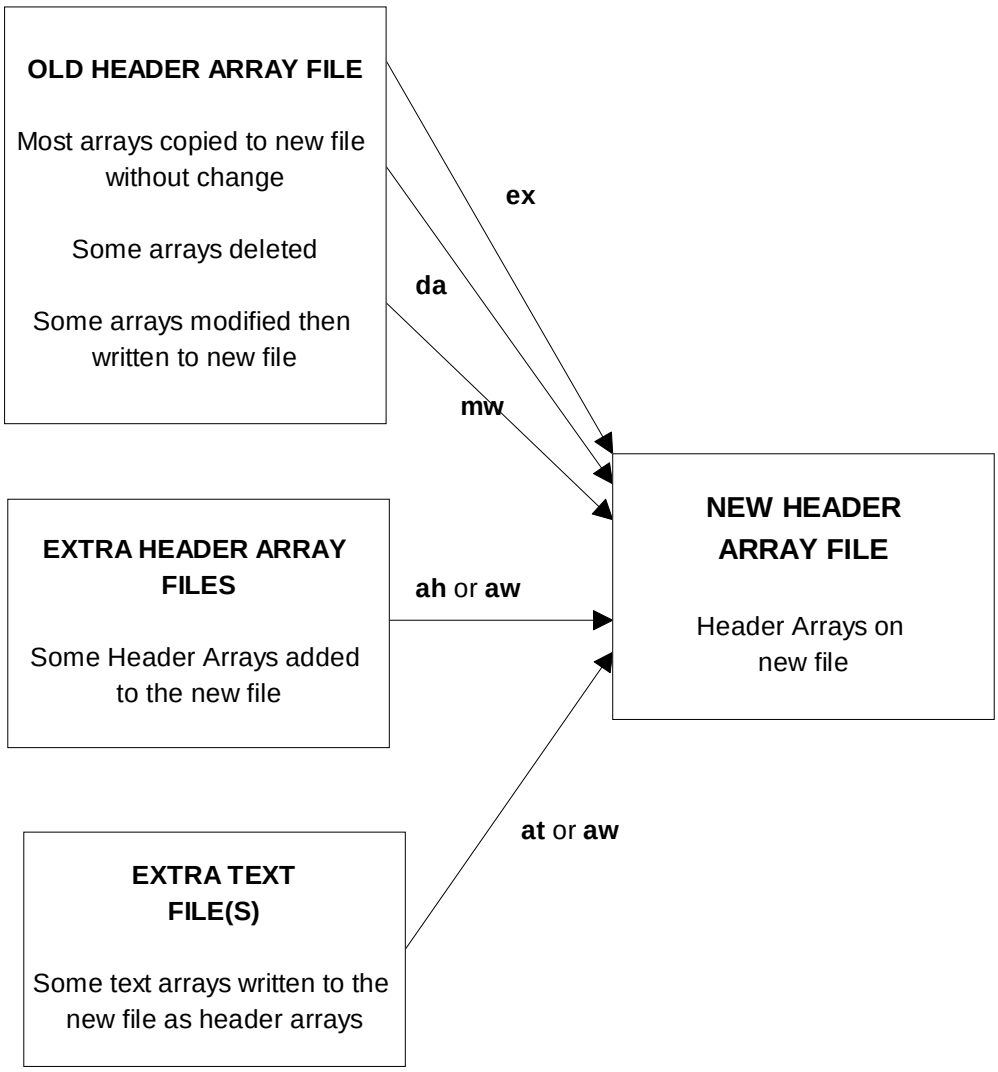


Figure 4: Commands for Modifying an Existing Header Array File

3.6.2 Commands for Adding Arrays to the New File

From an extra Header Array file

Menu Choice

ah	**add one or more arrays from a Header Array file**

Menu choice **ah** provides a simple procedure for adding complete arrays from an extra Header Array file. This extra file has arrays in header array format (so is a binary file). [Menu choice **at** provides a similar procedure for text files.]

After choosing **ah**, you will be asked for the name of the extra Header Array file. MODHAR displays the list of headers on the extra file and you are given the choice of adding

[a] all of the headers on the file

[L] some of the headers (list ones to add)

[m] most of the headers (list ones to omit)

[n] none of the headers.

For the choice [a] (which is the default here), the program adds ALL the arrays on the extra file to the new file. However, if the header of one of the arrays on the extra file is the same as that for an array already on the new file or for an array on the old file (but not deleted), the program will omit this array as adding it would cause duplicate headers on the new file.

For the choice [L], you supply a LIST of headers of arrays on the extra file to add to the new file. All other arrays on the extra file will be ignored. Again the program checks for duplicates and will tell you whether that header is already on the new or old file. If a duplicate is found, this array will not be added to the new file. Giving a carriage-return instead of a header ends the list of headers.

For the choice [m], the programs add MOST of the arrays on the extra files. You list the headers that you want to OMIT. A carriage-return ends the list.

For the choice [n], NONE of the arrays on the extra file are added. This gives a way out if you have chosen the wrong filename or if you just want to see what is on another file apart from the one you are modifying.

Example using the LIST option

The following example gives responses for adding arrays with headers 'EEEE' and 'FFFF' to the new file from an extra Header Array file.

User input

```
ah                  ! add one or more arrays from Header Array file
extra_header.har    ! name of extra file
L                   ! list ones to add
EEEE                ! header of array to add
FFFF                ! header of array to add
             ! carriage_return to end list to add
lh                  ! list headers
```

Headers 'EEEE' and 'FFFF' have been written to new file. The long names on the new file are the same as for these arrays on the extra file.

From a text file

Menu Choice

at	add one or more arrays from a text file

Menu choice **at** provides a simplified way of converting one or several arrays of data on a text file to header arrays and writing them to the new file in header array (that is, binary) form. [Menu choice **ah** provides a similar procedure for Header Array files.]

After choosing **at**, you supply the name of the extra text file and then are given the choice of adding

[a] all of the data arrays on the file

[s] some of the arrays

[n] none of the arrays

For the choice [a] (which is the default here), all of the arrays are added. If a valid header, and (optionally) a long name, are given in the "how much data" information (see chapter 6) for an array, MODHAR will convert the text arrays read in to header array format and write them to the new file with the given header.

If the header is not read in from the text file with the array, or is not a valid choice, or duplicates a header on the old or new file, you will be asked to provide a valid header for this array.

For the choice [s], MODHAR reads in the arrays one by one and displays the type, dimensions and, if given on the text file, the header and long name.

You have the choice of

1. accepting the header name with a carriage-return,
2. giving a different header for this array, (type in the new header eg ABCD)
3. skipping over this array (type XXX)
4. ending the reading of data from this file (type XXQ).

For 1 and 2, once a valid header is given, you are asked for the long name (if one is not specified in the text file) and then the array is written as a header array to the new file while for 3, no array is written. After this MODHAR goes on to read the next array till the end of file is reached or XXQ is input to end reading this text file.

For choice [n], none of the arrays are added and you are returned to the main menu.

Example using the SOME option

The following example gives responses for adding headers 'GGGG' and 'HHHH' to the new file from an extra text file.

User input

```
at                  ! add one or more arrays from text file
extra_text.txt      ! name of extra file
s                   ! add some of the arrays
              ! carriage_return to add 'GGGG'
HHHH          ! no header specified on the text file - input header
second array        ! long name for array
XXQ                 ! XXQ to end list to add
lh                  ! list headers.
```

Headers 'GGGG' and 'HHHH' have been written to new file.

The text file was as follows:

Text file

```
2  3  real  row_order  header  "GGGG"  longname "first array" ;
4.0  2.0  1.0
1.0  6.0  8.0
6  integer ;
1  2  3  4  5  6
3  strings  length  5  header  "xyza" ;
abcde
pqr
11223
```

The appropriate format for the text arrays on the text file is described fully in chapter 6.

Define Sets

Menu Choice

ds	**define sets for adding arrays from a text file**

Menu choice **ds** provides way to read in the element names in sets used in adding set and element information to arrays read in from a text file. A full example of this process is given in section 5.3.4.

The sole purpose of the **ds** command is to define certain set and element names which will then be used to add set and element labelling (see chapter 5) to arrays of data added from text files via the **at** command. See section 5.3.4 for more details about this.

You need to select **ds** and add the set information before you choose **at** to add the arrays from the text file. The names of the set elements are initially prepared on a Header Array file as arrays of character data.

On selecting the menu item ds, 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.

Note that the arrays used to define sets are not written explicitly to the new file. They are just stored in memory and used to add set and element labelling to other arrays.

3.7 Operations on Headers or Long Names

Initially MODHAR assumes that all arrays on the old file will be copied to the new file. To ensure headers on the new file are unique, MODHAR checks for duplicates in the new header list assuming

- all (non-deleted) arrays on the old file will be included on the new file.

Of course,

- headers of all arrays already written to the new file are included in the list of new headers.

(Initially, the "new" header associated with each array on the old file is its header on that file, but this can be changed using the **ch** command.)

Listing Headers

`Menu Choice`

lh	**list the headers, their status and size of associated arrays**
oh	**show one header, its status and size of associated array**

Menu option **lh** shows the complete list of headers while **oh** gives the same information for just one header. Status refers to whether the header is on the old file, the new file, an extra file or in memory during modification (**mw**). Old and new header names are given. In using **oh**, the header given should be the new header name of the array in question.

The following table is a sample of the output from **lh**.

No.	Header		Status	Data	Size		Size
	Old	New		type	old	new	change?
1	AAAA	ABCD	On old file	real	2 x 2	2 x 2	no
2	BBBB	BBBB	On old file	int	6	6	no
3	CCCC	CCCC	On old file	real	2 x 5	2 x 5	no
4	DDDD	DDDD	On old file	real	2	2	no
5	EEEE	EEEE	Deleted	int	3 x 4	3 x 4	no
6	FFFF	FFFF	Deleted	char	3 strs,len 5	3 strs,len 5	no
7	GAMM	GAMM	ON NEW FILE	real	1 x 2	1 x 2	no
8	PROD	PROD	ON NEW FILE	real	2 x 2	2 x 2	no
9	AAAA	AAAA	ON NEW FILE	int	6	6	no
10	FFFF	FFFF	ON NEW FILE	int	4	4	no
11	(TM)	GGGG	ON NEW FILE	real	2 x 4 x 7	2 x 4 x 7	no
12	CCCC	HHHH	IN MEMORY	real	2 x 5	6 x 5	YES

The original Header Array file contained arrays with headers AAAA, BBBB, CCCC, DDDD, EEEE, FFFF. Header AAAA was changed to ABCD using **ch** and arrays EEEE and FFFF were deleted using **da**.

Command **ah** and an extra Header Array file were used to add GAMM, PROD, and a new version of AAAA and FFFF.

Array GGGG was added from the terminal using command **aw** (see section 3.8). Command aw is currently being used to modify array CCCC from the old file to a new array with header HHHH; the size of this array has been increased from size 2 x 5 to 6 x 5.

Changing Header or Longname

Menu Choice

ch	**change a header name**
eln	**examine a long name**
cln	**change a long name associated with a header**

Menu option **ch** is used to change a header on the old file to a new different header. At the end when the array corresponding to this header is copied to the new file (see section 3.9), this array will have the new header associated with it.

Option **eln** is used to display the long name associated with a header and **cln** is used to change the long name to a new value. A “long name” associated with a Header Array is a character string of up to 70 characters which can be used to describe the array in more detail than the 4 character header.

Deleting or Restoring Arrays

Menu Choice

da	**delete one or more arrays from the old file**

To delete one or more arrays, choose the menu item **da**. You will be asked for a list of headers associated with arrays on the old file which you wish to delete. (These headers should be the names on the old file.) The associated arrays are marked 'deleted' and at the end of the program (when you exit) are not copied across to the new file. (They are, of course, not actually deleted from the old file, which is never changed by MODHAR.)

Only arrays on the old file can be deleted. (Arrays already written to the new file cannot be deleted.)

The header associated with a new array (added via **at**, **ah** or **aw** commands) cannot be the same as the new header associated with one of the un-deleted arrays on the old file. (This is because MODHAR must not allow duplicate headers on the new file.) In such a case you can either delete the old array using **da** or else change the new header associated with the old array using **ch**.

So, in summary, there are two reasons you might have for using **da**.

(i) To mark an old array as deleted so it will not be transferred to the new file when you exit MODHAR .

(ii) To clear the header name so you can associate it with another array of data.

Example of case (ii) above

For an old file containing arrays with headers 'AAAA', 'BBBB', 'CCCC', we wish to remove the old arrays 'AAAA' and 'CCCC' and insert from an extra Header Array file, new arrays with headers 'AAAA', and 'CCCC'. After opening the old and new files, give the following responses.

User input

```
da           ! delete one or more arrays from the old file
AAAA         ! header on old file to delete
CCCC         ! next header to delete
       ! carriage_return to end the list
ah           ! add one or more arrays from Header Array file
extra_header.har ! name of extra file
L            ! list ones to add
AAAA         ! header on extra file to add
CCCC         ! next header to add
       ! carriage return to end the list
lh           ! list headers to see the result
```

Menu Choice

ra	**restore one or more deleted arrays on the old file**

If an array has been deleted from the old file, you can restore it using the menu choice **ra**. You will be asked for a list of headers to restore to the list of headers of arrays to be copied across to the new file from the old file on exit.

As with all cases when a header is added to the list of headers to be added to the new file, MODHAR checks for duplicates which might occur on the new file (including those to be copied from the old file). If the program finds a possible duplicate, you will be asked to supply a new header for the array you are restoring.

3.8 Commands for Complicated Modification or Addition

Modify Write or Add Write

Menu Choice

mw	**modify data from the old file and write it on the new file**
aw	**add new data and write it to the file**

If you choose either **mw** or **aw**, a (secondary) MODHAR Sub-command Menu becomes active, offering the Sub-commands shown below.

Note that you should use command **aw** rather than **at** or **ah** if

- you want to modify the data read before writing it to the new file, or
- you want to enter the new data from the terminal.

3.8.1 MODHAR Sub-commands for *mw* or *aw*

m	modify the data (before writing it)
ed	examine part of the data (held in memory)
is	increase the size of the array
w	write (all of) the array to the new file
wp	write part of the array to the new file
a	abort this Sub-command, and go to the Command menu
lh	list the headers, their status and size of associated arrays
oh	show one header, its status and size of associated array
ch	change a header name
eln	examine a long name
cln	change the long name associated with a header
?	help

For users unfamiliar with the operation of MODHAR, use of the help command ? lists the current menu to the terminal screen. The use of the command **lh** (available in both menus) to list the current headers on both the original, old Header Array file (if one is being used) and the new Header Array file will also help clarify MODHAR's operation to new users.

MODHAR gives a range of prompts depending on which Subcommand has been chosen. Generally, you name the header of an array to be modified. MODHAR reads the array into a work array in

memory. You can then examine the data in this array **ed**, increase its size **is** or modify the data **m** by replacing parts of it or scaling its values.

When the modification of the array is complete, it is written to the new file using **w** or **wp**. Once an array is written to the new file it cannot be further modified.

3.8.2 Examples

Some examples may be useful to indicate the operation of MODHAR and the types of questions MODHAR may ask the user.

Source of data *Example 1*

When you want to add an array to the new Header Array file, a series of MODHAR prompts will ask you where the array data comes from. The possibilities are :

- directly via input by the user at the terminal,
- from a formatted text file,
- from the old Header Array file upon which the new Header Array file is based or
- from some other Header Array file.

Terminal Input *Example 2*

When you are typing in the actual values for a new array to be added to the new Header Array file, MODHAR will ask a series of questions :

- what type of data is being added - real, integer or character values
- the number of dimensions of data
- the actual dimension sizes
- the values for the array data are prompted for, with a clear indication of their position in the array. For example, for a 6x3x4 real array, you will receive prompts of the form:

Please enter the 6 values for positions (1-6, 2, 3)

Modifying the data *Example 3*

When modifying an array that has been just read by MODHAR (including an array whose values have just been input by the user), MODHAR will ask such questions as:

- how much of the array is to be modified ? - possibilities are one single entry, a few entries, a rectangular sub-array or the whole array
- is the modification a scaling of the current values, an addition to the current values, or a replacement?
- are all entries to be modified in the same way ? i.e. either scale all values by the same factor, or add the same amount to all values, or replace all values with the same replacement value
- if the modification is different for different values, are the individual modification (scaling, adding or replacement) values read from the terminal, from a text file, from the old Header Array file upon which the new Header Array file is based or from some other Header Array file ?

Modifying a sub-array on an old Header Array file *Example 4*

This example gives full responses for a run of MODHAR which

- Modifies an array on the old file before writing it to the new file
- First the array dimensions are increased from 2 x 3 to 4 x 4.
  (The new positions initially contain zero values.)
- Then two new data values are added as a subarray in positions (3–>4,4) from data input from the terminal

The old array 'abcd' was

4.0 2.0 1.0

1.0 6.0 8.0

After modification the new array will be

4.0 2.0 1.0 0.0

1.0 6.0 8.0 0.0

0.0 0.0 0.0 3.0

0.0 0.0 0.0 5.0

User input

```
<carriage-return>   ! carriage return to end option selection
y                   ! based on an old Header Array file
old_file.har        ! Name of old Header Array file
new_file.har        ! Name of new file
lh                  ! list headers of arrays on old file
mw                  ! modify an array on old file and write it to new file
abcd                ! header of array to be modified
is                  ! increase size of array
1 2                 ! increase dimension 1 by 2, that is, from 2 to 4
2 1                 ! increase dimension 2 by 1, that is, from 3 to 4
end                 ! end selection of dimensions to increase
m                   ! modify this array before writing to new file
r                   ! replace some parts of the array data
s                   ! modify a subarray
1 3 4               ! select entries 3 to 4 in first dimension
2 4 4               ! select entry 4 in second dimension
end                 ! end of selection of subarray (3-4,4) of size 2 x 1
g                   ! general matrix used to modify array
t                   ! input modifying matrix from terminal
3.0 5.0             ! enter 2 values since array is of size 2 x 1
?                   ! view menu for list of commands
w                   ! write all the array to the new file
n                   ! don't want to reuse data in memory
ex                  ! exit saving the new header array file
a                   ! all remaining arrays to be transferred
=                   ! use old history without adding to it
```

3.8.3 Writing Arrays to the New Header Array File

Once you have completed all modifications to an array, the array can be written to the new Header Array file. MODHAR gives you the choice of writing either the entire array (using the **w** Command) or only a rectangular sub-array of the entire array (using the **wp** Command). Before proceeding onto another array, you can also base an entirely new Array on the most recently written array. In this way, any complicated modifications just performed for that array need not be repeated.

Note that, although there is an **is** Subcommand for increasing the size of an array, no ds ("Decrease the size") Subcommand is needed since you can decrease the size by only writing part of the array.

Writing part of an array using wp *Example 5*

This example gives full responses for a run of MODHAR which

- examines part of an array using the **ed** command
- writes part of an array to the new file using the **wp** command.

Initially the array has dimensions 4 x 3 x 4 x 5 x 2.

On the new file, the sub-array written is

1 , 1 -> 3 , 2 , 2 -> 5 , 1 -> 2

The new array has dimensions 1 x 3 x 1 x 4 x 2.

User input

```
<carriage-return>   ! carriage return to end option selection
y                         ! based on old file
old.har                   ! old file name
new.har                   ! new file name
lh                        ! list headers on old file
mw                        ! choose modify and write option
JJJJ                      ! header of array to be modified
ed                        ! examine data
s                         ! examine a submatrix
2 1 3               ! second dimension elements 1-->3
3 2 2               ! third dimension element 2
4 2 5               ! fourth dimension elements 2-->5
5 1 2               ! fifth dimension elements 1-->2
end                       ! end of selection of submatrix
?                         ! list menu of options
wp                        ! write part of the array to the new file
2 1 3               ! write second dimension elements 1-->3
3 2 2               ! third dimension element 2
4 2 5               ! write fourth dimension elements 2-->5
5 1 2               ! write fifth dimension elements 1-->2
end          ! end selection - written 1 x 3 x 1 x 4 x 2 array
n                         ! don't reuse data in memory
lh                        ! list headers
ex                        ! exit saving the new file
a                         ! all remaining arrays to be transferred
Jill                      ! name as part of history
12 -1-00                  ! date (Only prompted for on some machines)
Example of selecting part of an array
**end
y                         ! accept history for new file
```

3.8.4 Option ADD when Modifying Data in MODHAR

Prior to Release 6.0-001, when you modify the data on an array using MODHAR, you were offered the options:
"r" (replace) or "s" (scale).

MODHAR now[9] also offers the new alternative **"a" (add)**.

With this option, the numbers entered are added to the values at the original header. Of course you can subtract values by adding negative values. (For example, if you add -2 to each entry of a matrix, this is the same as subtracting 2 from each entry.)

3.9 Finishing Up

If the new Header Array file has been based on an existing, Header Array file, then when you have finished either adding any new arrays or writing any modified arrays to the new Header Array file, MODHAR gives you the option of copying, unchanged, some or all of the arrays from the old file that have not yet been transferred to the new file. So, if you wish to copy an existing Header Array file containing 100 arrays, only one of which needs to be modified, then you only need to access explicitly the array to be modified. Once the single modified array has been written to the new Header Array file and you enter the **ex** command to exit MODHAR, MODHAR will ask if you want to copy some or all of the untouched 99 arrays from the old Header Array file across to the new Header Array file. In this way, you are saved from having to access explicitly any array that requires no modifications.

3.9.1 Exit Commands

Exit or Quit

```
Menu Choice
```

ex	**exit, saving the new Header Array file**
q	**quit without saving the new Header Array file**

When you have finished adding, deleting and modifying arrays, menu choice **ex** is the way to exit MODHAR if you want to save the newly created Header Array file. Choice **q** aborts the process and deletes the new file.

If the new Header Array file has been based on an existing Header Array file and you give command **ex**, MODHAR gives you the option of copying unchanged some or all of the arrays from the old file which have not previously been transferred to the new file and have not been deleted using **da**. The choices are:

[a] ALL remaining arrays to be transferred

[L] LIST headers of arrays to be transferred

[m] MOST of the arrays (List those not transferred)

[p] Respond to PROMPTS to enter which to transfer

[n] NONE of the arrays to be transferred.

[a], [L], [m], and [n] are similar to those in menu item **ah**.

For the choice [a], the program adds ALL the non-deleted arrays on the old file to the new file.

---

[9] This was new for Release 6.0-001.

For the choice [L], you supply a LIST of headers of arrays on the old file to add to the new file. All other arrays on the old file will be omitted. Giving a carriage-return instead of a header ends the list of headers.

For the choice [m], MODHAR adds MOST of the arrays on the old file. You list the headers that you want to OMIT. A carriage-return ends the list.

For the choice [n], NONE of the remaining arrays on the old file are added.

For the choice [p], MODHAR prompts you with the header names of undeleted arrays on the old file and you choose from the following list of actions:

[y] Transfer this array to the new file.

[n] Do NOT transfer this array.

[r] Transfer this and ALL REMAINING arrays

[q] Do NOT transfer this or ANY REMAINING ARRAYS.

[?] Display this menu again.

After transferring the requested arrays, a list of headers is produced showing the status and sizes of the arrays on the old and new files.

3.9.2 History of the New File

Finally you are asked to add to the history by supplying your name and any notes you wish to append to the current history about what changes have been made in this run of MODHAR. (On some machines you may also be asked to input the date.)

If you don't want to add to the history in this way, you can choose from

[-] ignore any old history. (Start the history with a fresh slate - type in your name and a new history.)

[=] to take any old history (no new). (Copy the old history to the new file without adding to it.)

[0] to have no history on the new file.

3.10 Complete Example of a MODHAR Run

The following example gives all responses for a MODHAR run. If you want to try it yourself, create the following three files (either using option **at** and taking data from a text file, or using option **aw** and giving input from the terminal.)

Initially you need

(1) An old file ‘old_file.har’ containing arrays with headers ‘AAAA’, ‘abcd’, ‘EEEE’ and ‘FFFF’

(2) An extra file ‘moupd.har’ containing arrays with headers ‘prod’ and ‘gamm’

(3) A second extra file ‘extra_header.har’ containing an array with header ‘AAAA’.

User input

```
<carriage-return>   ! carriage return to end option selection
y                   ! based on an old file
old_file.har        ! name of old file
new_file.har        ! name of new file
lh                  ! list headers on old file
ah                  ! add arrays from extra Header Array file
moupd.har           ! name of extra file
L                   ! list headers of arrays to add
prod
gamm
<carriage-return>   ! carriage return to end list
lh                  ! list headers to see 'prod' and 'gamm' on new file
da                  ! delete arrays on old file
AAAA                ! header of array to delete
FFFF                ! header of array to delete
<carriage-return>   ! carriage return to end list
lh     ! list headers to see deleted arrays 'AAAA' and 'FFFF' on old file
ah                  ! add arrays from an extra Header Array file
extra_header.har    ! name of extra file
L                   ! list headers to add to new file
AAAA         ! header to transfer (permitted since old 'AAAA' is deleted)
<carriage-return>   ! carriage return to end list
lh
ra                  ! restore array deleted from the old file
AAAA                ! this is the original header on the old file
BBBB   ! necessary to enter a new header because 'AAAA' is on new file
<carriage-return>   ! carriage return to end list to restore
lh
aw                  ! add a new array and write it to the new file
o                   ! from the old file
cdef                ! header of array on old file
fghi         ! 'cdef' is on old file so a new header is needed 'fghi'
w                   ! write array to new file
n                   ! don't reuse array in memory
lh
ex                  ! exit saving the new file
a      ! all remaining (non-deleted) arrays to be copied to new file
Jill   ! add to history on new file
29/9/99 ! Date (Only prompted for on some machines)
Example for MODHAR
**end               ! end of history
y                   ! confirm history before adding
```

After running this example, on the new file are arrays with headers 'BBBB' (was 'AAAA' on old file), 'abcd', 'cdef', 'EEEE'. All these were originally on the old file.

Also on the new file are arrays 'gamm' and 'prod', and the new version of 'AAAA' from extra Header Array files.

The new file also contains a second copy of array 'cdef' from the old file now renamed 'fghi'.

3.11 Text Files

MODHAR allows the use of prepared text files to input a variety of data, the data values of an array, the scaling factors to be applied to values in an array, or the replacement values for arrays or subarrays.

A text file is a file which can be viewed or edited with a text editor. Text files which GEMPACK programs can read have a particular format and are referred to as GEMPACK Text files. Full details of preparing arrays for GEMPACK Text files are given in chapter 6.

In MODHAR when using option choice **at**, the GEMPACK text file can contain more than one array of data. In other parts of MODHAR, only one array should be given on each text file.

CHAPTER 4

4. Viewing or Comparing Header Array Files

This chapter describes how you can get information about the data on a Header Array file. Header Array files are often briefly referred to as HAR files. The relevant programs are

ViewHAR	View the data on a Header Array file, discussed in section 2.2
SEEHAR	See the data on a Header Array file, see section 4.1
CMPHAR	Compare the data on two Header Array files, see section 4.2

All 3 are easy to run interactively. ViewHAR can only be run on a Windows PC.

There are other programs which operate on Header Array files:

SUMHAR	Summarise the data on a Header Array file, see section 4.3
RWHAR	Read a HAR file and write the data as a formatted (text) file
MKHAR	Read the output by RWHAR and convert it to a Header Array file
MODHAR	Modify the data on Header Array files, discussed in chapter 3
CMBHAR	Combining similar Header Array files, see section 4.4

RWHAR and MKHAR are used in transferring data between different computers, for example, Windows PC to Unix computer, and will be discussed in chapter 11.

4.1 SEEHAR: Prepare a Print File or CSV Spreadsheet File

On a Windows PC,

> SEEHAR can be run using WinGEM by selecting ***HA Files | See SEEHAR*** or interactively at the Command prompt. To select options in WinGEM, in the SEEHAR window, select
>
> ***Options | SEEHAR Options***
>
> and then click ***Ok*** to return to the SEEHAR window.
>
> ***Select*** the Header Array file you wish to look at, and then ***Run*** SEEHAR.

Alternatively you can run SEEHAR on any machine by typing SEEHAR on the command line (which means in a DOS box on a Windows PC).

You can run SEEHAR to send to a Print file the actual data associated with some or all of the headers on a Header Array file. (Then you can print the Print file or view it in a text editor if you wish.) You can choose one of 3 forms of output.

1) If no special options are chosen, the default is to have row and column labels which indicate the position of each piece of data in the whole array. We refer to this as **labelled output from SEEHAR**.

   Labelled output of real arrays has exactly the same form as the output produced by DISPLAY statements as written by GEMSIM and TABLO-generated programs (see section 4.3 of GPD-3).

2) An alternative is to have the output in a form which allows you to import the arrays into a spreadsheet easily.[10]
The separator between the data entries can be chosen to be a comma (the default) or a TAB character (or any other desired character). To obtain this type of output, select the SEEHAR option

SS Spreadsheet output, no element labels

or

SSE Spreadsheet with element labels

at the start of the program, after which you will be asked which separator you want. Most spreadsheet programs (on PCs, for example) will accept as input the kind of file produced. If you choose a comma as the separator, the output will be in what these spreadsheet programs call **comma-separated values** or **CSV** format.
Option SSE puts element labels on the output (see section 5.1.2 for an example of SSE output), whereas option SS does not. Option SSE is aimed at the production of reports whereas option SS can be used when you want to read data into a spreadsheet, modify it there and then read it back into a Header Array file.

3) Another alternative is for the output to be a GEMPACK text data file, from which the data can be read easily into other GEMPACK programs. Specifically, each array is output in the form required for text input to MODHAR, GEMSIM or to TABLO-generated programs (as documented in chapter 6). To obtain this type of output, select the SEEHAR option

ROW Unlabelled output (row order)

or

COL Unlabelled output (column order)

at the start of the program. Arrays containing real or integer data will be output in row or column order respectively, at most 4 or 5 entries per line of the file. Arrays with character data will be output with each character string on a new line. With real or integer output, element name or number labels will make it clear where each number is placed in the array (though you can select option 'NEL' to suppress this labelling if you wish).

To see how SEEHAR works, you could use it (choose the default options) to print all the data on the data file (probably called SJ.DAT) for Stylized Johansen.

In all cases above, a three or higher dimensional array is output as several matrices.

For example, a 3 x 10 x 6 x 4 array is output as 24 (=6x4) matrices each of size 3 x 10. In case (1) above, the labels specify clearly the position of each piece of data in the whole array. In case (2) above, the entries in each constituent matrix are in row order. In case (3) above, each matrix is output in row or column order and each new row or column starts on a new line. (For row order, each matrix will be output as 3 rows each containing 10 numbers, while for column order, each matrix will be output as 10 columns each containing 3 numbers.)

In cases (2) and (3) above, if several arrays of data are on the output file, the header and long name associated with each array are shown at the start of each array. The header and long name are written using the syntax described in section 6.1.4. In some cases you may need to split the output file into separate arrays (using a text editor) before using it as input into a spreadsheet or another GEMPACK program.

---

[10] We are grateful to Kevin Hanslow for suggesting spreadsheet output and for providing us with a prototype implementation of this.

4.1.1 SEEHAR - Options

SEEHAR has several options including

CPW Change page width

CPL Change the page length

DEC Number of decimals. This is only relevant for labelled output. The default is to show 6 figures after the decimal place for real numbers. Use this option if you want fewer.[11]

FNP Few new pages which instructs SEEHAR to start a new page only when required for labelled output (the default output from SEEHAR).

4.1.2 Displays and Labelled Output from SEEHAR into Spreadsheets

Christian Bach has pointed out that the output from DISPLAYs in your TABLO Input file 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.

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 – see section 4.3.1 of GPD-3) or via SEEHAR's option CPW (Change Page Width – see section 4.1.1).

See also chapter 9 for several ways of making tables in spreadsheets using SLTOHT.

4.1.3 GAMS Output from SEEHAR, TABLO-generated Programs and GEMSIM

SEEHAR can also be used to translate a Header Array file to GAMS format (see chapter 16) by selecting option **GMS.**

The output contains GAMS-type lines at the start and end of the files.

(i) The lines at the start is to make sure that GAMS treats the GEMPACK-style comment lines as comments. This saves users the trouble of removing these lines.

(ii) The lines at the end attempt to return the status at the end of the file to something close to what it was (as far as GAMS is concerned) at the start of the file.

These are designed to make it easier for these files to be joined together with other GAMS files.

[11] Prior to Release 5.2, when producing labelled output, SEEHAR asked you how many figures to put after the decimal point. This question is no longer asked.

4.1.4 Converting Header Array Files to Data Bases - Option SQL

The option **SQL** in SEEHAR enables it to produce output which makes it relatively easy to add the data in the file to various databases.[12] We are grateful to Frank van Tongeren for suggesting this feature.

If option SQL is chosen, the Header Array file data is converted into SQL statements. These SQL statements can be implemented on a database system to create tables in a database and load the CGE model data into the tables. Currently the only type of arrays which SEEHAR converts to SQL statements are RE arrays (that is, arrays of real numbers with associated set and element labelling – see section 5). The other types of arrays are ignored.

The database created has 3 tables:

1. The values in real arrays on the Header array file are stored in the table RARRAY.
2. The table SETLIST is a list of sets and their elements for each Header Array.
3. The table HEADLIST is a table of the headers on the file with various properties related to that particular Header.

The database tables are initially created by a CREATE TABLE statement, then this is followed by a large number of INSERT statements, one for every real value in every array.

The SQL statements are written in a text file (separated by semi-colons).

To see an example of the output, run SEEHAR and choose the option SQL. Either select the Headers one by one 'o', or output all Headers on the file 'r'. You can choose a different table name for each Header if you output the tables one at a time using the ‘o’ option. If you use the ‘r’ option all real values are inserted into the table RARRAY.

In order to use this SQL text file on a database system, the first step is to create a database, a process which varies according to the database system. Once you have a database, you can connect to it and run the SQL text file as a script to insert all the values on the Header Array file into the database.

> For example, in Oracle, you can use the SQLPlus program to carry out the loading. In the Windows version of Oracle, at the command prompt, to run the SQL file TEST.SQL for example, enter
>
> > plus80 user/password@databasename @TEST.SQL
>
> This process can be fairly slow if dealing with a long file. However once completed you have an Oracle database, to query in whatever way you like.

Delphi can also be used to run the SQL text file to insert the values into a database, for example, a database for MS Access.

[12] The option SQL was first introduced in Release 7.0. SQL, which stands for “Structured Query Language”, is data base terminology which is widely used.

4.2 CMPHAR: Comparing Data on Header Array Files

Run the program CMPHAR if you wish to compare the data at the same headers on 2 different Header Array files. The output is sent to a text file.

For each header which is on both files, CMPHAR compares the data entry by entry if the arrays are of the same size (but ignores these arrays if they are of different size). Arrays associated with a header which is only on one of the files are ignored by CMPHAR.

When two arrays are compared, they are compared entry by entry. For two entries a1 (on file 1) and a2 (on file 2) in the same position in the array, CMPHAR calculates and reports

1) the **absolute difference** |a1-a2|,

2) (provided a1 and a2 are both different from zero), the **difference ratio** defined by

```
|a1-a2| / MIN(|a1|,|a2|).
```

When the two entries are close together, the absolute difference and difference ratio will be close to zero. Two very large numbers (say in the hundreds or thousands of millions) may be relatively close together but still have a "large" absolute difference. For example, consider the values 210123456.6 and 210122456.6. Their absolute difference is 1000 but the associated difference ratio is only about 0.000005 which is quite small. In such cases, difference ratios may be more meaningful measures as to how "close" the two lots of data are than absolute differences. (On the other hand, two small numbers can be very close but have a large difference ratio, such as the numbers 0.000001 and 0.000002 whose difference ratio is 1.)

4.2.1 CMPHAR Report

For each pair of arrays compared, if their data is not identical, CMPHAR reports

1) the total of all absolute differences, the average of these, the largest one and the position in the array and the respective values where this maximum occurs, and

2) the total of all difference ratios, the average of these, the largest one and the position in the array and the respective values where this maximum occurs.

It also gives the same information totalled and averaged across all arrays compared on the two files.

4.2.2 Significant Differences

Note that very small differences in the way two lots of data are calculated can make small differences in the numerical results obtained (even if the two different ways of doing the calculation should theoretically give the same answers). This is because of rounding errors which are inevitable on computers. For example, small changes such as different condensations of the model (or even condensation actions carried out in a different order) can make small changes in simulation results and in the updated data files. On many machines, difference ratios less than 0.000001 or even 0.00001 can be thought of as insignificant because of these considerations.

CMPHAR normally reports all differences. However, via option '**s**' in the main menu, you can tell it to report only "significant" differences, that is, places where the absolute difference exceeds a number you nominate (for example, 0.00001) and the difference ratio exceeds a second number you nominate (for example, 0.00005).

4.2.3 Comparing Solution Files

CMPHAR can also be used to compare the values on two Solution files from the same model. To do this, you must first convert the Solution files to Header Array files using the program SLTOHT (see chapter 8). Then select CMPHAR option

SOL Compare solution files

when starting the run. You will need to give CMPHAR the names of the original Solution files as well as of the Header Array versions produced by SLTOHT. When used in this way, CMPHAR reports summaries of the differences in the numerical results for each variable in turn.

4.3 SUMHAR: Summarizing a Header Array File

Run SUMHAR to summarise the data on a Header Array file. For each array of data on the file, this tells you the associated header, the size of the array and whether the data is real, integer or character. The output from SUMHAR can be sent to the terminal or to a Print file or to both.
To see how SUMHAR works, you could use it to obtain a summary of the data file (probably called SJ.DAT) for Stylized Johansen.

4.4 CMBHAR: Combining similar Header Array Files

Run CMBHAR to combine the data on several Header Array files. The Header Array files must have just the same Headers on each of them. CMBHAR will ask you for the number of Header Array files and their names, and the name of the output file.

CMBHAR adds another array dimension to each array with elements in the set Combined. The element names of this combined set are the names of the Header Array files (without suffix) which have been read in.

To see how CMBHAR works, you could use it to combine the data file (probably called SJ.DAT) for Stylized Johansen and the updated data file SJLB.UPD. The set Combined in this case is (SJ, SJLB). Look at the output in ViewHAR or SEEHAR to see the result.

CHAPTER 5

5. Set and Element Labelling on Header Array Files

Arrays of real numbers on Header Array files can contain set and element labelling information[13], 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 ViewHAR (see section 2.2).

Below is an example of labelled output from SEEHAR (see section 4.1) 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.

Note that set and element labelling information can only be attached to arrays of real numbers on a Header Array file. There is never set and element information attached to arrays of integers or character strings on a Header Array file.

This chapter describes how to put this information onto an existing (or new) Header Array file and explains various related consequences for GEMPACK programs.

[13] 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.

5.1 Effect of Set and Element Information on GEMPACK Programs

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

As noted in section 3.1.1 above, the type of each array is shown when SEEHAR or ViewHAR lists 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 9 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 9.3).

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 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 section 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.

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.

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 check whether this set/element/coefficient information as held on the file agrees with what they expect. For details see section 4.4 of GPD-3.

We realise that there may be circumstances when you do not want such checking to occur. If so, it is easy to suppress part or all of this checking. Details can be found in section 4.4 of GPD-3.

5.3 Adding Set and Element Information to an Existing HAR 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 4.6.6 and 4.6.7 in GPD-2) was new for Release 6.0. Method 5 uses ViewHAR.

5.3.1 Method 1 - Extra Write Statements

One way of adding the set and element information to an existing Header Array file, is to add **xwrite** statements (see section 6.6 of GPD-3) 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 2.8.1 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.

Note that you do not need to include the LONGNAME in the XWRITE statements since long names are automatically transferred (see section 4.10.6 of GPD-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 uses MODHAR and relies on the TABLO features described in sections 4.6.6 and 4.6.7 in GPD-2.

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 6. Alternatively you can use a WRITE(ALLSETS) or XWRITE(ALLSETS) statement (see section 4.6.7 in GPD-2) 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 section 6.1.5)**.** 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 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 4.6.7 in GPD-2)

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

   You need to define the sets prior to referring to them in step 3.[14]

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
```

[14] The sole purpose of the **ds** command is to define certain set and element names which will then be used to add set and element labelling to arrays of data added from text files via the **at** command. See the documentation of the 'ds' command in section 3.6.2 for more details.

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).

5.3.5 Method 5 - Using ViewHAR

Using the Full Editing menu in ViewHAR, select ***Edit | Apply/Change Set Labels*** menu command to add or modify set labelling information for real matrices.

Edit one dimension at a time. You will be offered a list of all sets in the Set Library of the appropriate size. Or, choose "No Set" to remove set labelling information.

The easiest way to fill the set library with a useful selection of sets is to open, then close, a HAR file containing a number of labelled real matrices. Then open the HAR file containing un-labelled real matrices and apply set labels to each matrix in turn.

Further details are given in the ViewHAR on-line help.

5.4 Compatibility 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 or later programs. (If they attempt it, they will almost certainly crash with an error.)

To enable Release 5.2 and later 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 (a Release 5.2 or later 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).

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. Syntax of GEMPACK Text Data Files

This contains complete and self-contained documentation for GEMPACK text data files.[15]

This chapter is relevant for many different GEMPACK programs.

- GEMPACK text data files can be read or written by GEMSIM and TABLO-generated programs (see, for example, section 4.9.1 of GPD-2).
- They can be used as shocks files for GEMSIM and TABLO-generated programs (see section 5.8 of GPD-3).
- Text data files can be written by SEEHAR (section 4.1) and SLTOHT (see, for example, sections 8.1 and 8.7).
- Text data files can be read by MODHAR (see, for example, sections 3.6.2 and 3.11).
- GEMPACK text data files can be read or written by ViewHAR (see section 2.2).

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.

6.1 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.

6.1.1 Real or Integer Data

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).

---

[15] 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.

In all cases, real, integer or character, a **semicolon ;** is required at the end of the "how much data" information.

In general, GEMPACK programs that are reading or writing the “how much data” information, expect the information in the order given above: {sizes, type, order,.....}. However the coefficient information may be read or written at the beginning of the “how much data” information. For example, the program SEEHAR usually writes the coefficient information (if available) at the start of the “how much data” information rather than the end.

6.1.2 Examples of "How Much Data" Information

(1) For a real vector of size 6, the "how much data" information is

```
6 ;
```

(2) 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 ;
```

or

```
2  3  4  ;
```

(3) For an vector (that is, a 1-dimensional array) containing integers of size 7, the "how much data" information is

```
7  integer  header  "IJK9" ;
```

(4) 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 " ;
```

6.1.3 Character String Data

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. Again <header> and <longname> are optional (see below).

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" ;
```

6.1.4 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' – see sections 3.6.2 and 3.8). 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.

6.1.5 Coefficient

The final term[16] 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 (see section 3.6.2). 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 (see section 3.6.2), 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.

6.1.6 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.

6.1.7 Complete Example of GEMPACK Text File

Below is an example of a GEMPACK Text file containing 6 arrays of various type.

[16] Alternatively the coefficient information may be read or written at the beginning of the “how much data” information. For example the program SEEHAR usually writes the coefficient information (if available) at the start of the “how much data” information rather than the end.

```
!         A complete example text file containing 6 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
1w223

! The coefficient information can come first

coefficient DVCOMIN(SECT,SECT)
2 2 header “CINP” longname
“Intermediate inputs to commodities” ;
4.0  2.0
2.0  6.0
```

6.2 Data Values in Text Files

6.2.1 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.

6.2.2 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
```

6.2.3 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.

6.2.4 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.

6.2.5 Spreadsheet Order

- 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.

6.2.6 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

Do not leave a space after the “-“ or “+” sign. (For example, do not put “- 15.9”.)

6.2.7 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

(no space after the "-").

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

6.2.8 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.

6.2.9 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.

6.2.10 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.

6.2.11 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 6.1.6 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.

6.2.12 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.

6.2.13 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.

6.2.14 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 6.1.6 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
```

6.3 Slight Incompatibility Between Release 5.2 and 6.0 Text Files

Release 6.0 (and later) programs including SEEHAR write the coefficient name as part of the "how much data" heading at the start of each array. This coefficient name cannot be read by Release 5.2 GEMPACK programs.

CHAPTER 7

7. GEMPIE: Printing Simulation Variable Results

Solution files are binary files produced by GEMSIM, TABLO-generated programs and SAGEM (see chapter 8 of GEMPACK document GPD-3). The programs used to process Solution files are ViewSOL (see section 2.3), GEMPIE and SLTOHT (see chapters 8 and 9).

The standard use of GEMPIE to make a Print file was introduced in chapter 2 of GPD-1. In this chapter we give more details about using GEMPIE to access results on a Solution file.

When reading this chapter, you need to be clear about the different sorts of solutions that can be held on a Solution file – see section 8.2 of GPD-3. In particular, recall that Solution files produced by GEMSIM and TABLO-generated programs never contain individual column results but contain the totals (or cumulative) results and may also contain levels and/or subtotals results. Solution files produced by SAGEM never contain levels results but may contain the totals results, subtotals results and/or individual column results.

Section 7.1 describes how to run GEMPIE via WinGEM. Sections 7.2 and 7.3 give information about running GEMPIE interactively. The remaining sections apply however GEMPIE is run.

7.1 Using WinGEM to Run GEMPIE

Select ***Simulation | GEMPIE Print*** from WinGEM's main menu. If you are running a simulation, you may be able to select the *Go To GEMPIE* button as in section 2.5.5.3 in GPD-1.

Select the Solution file you wish to view. Click on ***Solutions?*** to see what solutions are available for printing. Usually after you have chosen the Solution file , WinGEM will ask you which solutions you want to print if there is a relevant choice. Possible choices are:

- Individual columns solutions
- The Totals solutions
- Totals and subtotals solutions
- Totals and levels solutions

[If any of these 4 choices is not available the choice will be greyed out.]

Click ***Ok*** to return to the GEMPIE window, ***Run*** the program and ***View*** or ***Print*** the results.

If you need to choose special Options for GEMPIE, select

Options | Gempie Options

before running the program.

7.2 Using GEMPIE Interactively

In this section we give details about running GEMPIE interactively. Even if you normally run GEMPIE via WinGEM, you may need to run GEMPIE interactively. Run GEMPIE interactively if you wish to use GEMPIE's capabilities of selecting lists of variables (for example, you only want to print the results for some of the endogenous variables – see section 7.3), or if you wish to set up new subtotals from individual column results produced by SAGEM (see section 7.2.2).

We consider Solution files produced by GEMSIM and TABLO-generated programs separately from those produced by SAGEM since the files can contain different sorts of solutions.

7.2.1 Solution Files Produced by GEMSIM or TABLO-generated Programs

Recall from section 8.2 of GPD-3 that Solution files produced by GEMSIM and TABLO-generated programs never contain individual column results but contain the totals (or cumulative) results and may also contain levels and/or subtotals results.

GEMPIE can process and show levels results and subtotals results but not both in the same run. In any one run of GEMPIE you can print

- either the cumulative solution plus any subtotals,
- or the cumulative solution plus any levels results.

When you run GEMPIE interactively, after the options choice, you specify the name of the Solution file you wish to access.

Then you are asked to choose which cumulatively-retained endogenous variables you wish to be printed. The simplest choice is 'a' which means all. See section 7.3 below for advice about making other choices. What happens next depends on what types of solutions are on the file.

(1) If the Solution file you are accessing contains both levels results and subtotals results,[17] your next choice is to indicate which of these you wish to print.

```
There are Levels results and also Subtotal results on this file.

Type L to print Levels results
     s to print Totals/Subtotals results.
```

If you select "L" for levels results, that is the end of the interesting choices.

If you select "s" for totals/subtotals, you are then asked to indicate which (if any) subtotals you want.

(2) If the Solution file you are accessing contains subtotals results but no levels results, your next choice is to indicate which (if any) subtotals you want.

(3) If the Solution file you are accessing contains levels results but no subtotals results, you automatically see the levels results and the totals results unless you selected option **NLV** (No levels results) from the GEMPIE options menu, in which case you just see the totals results.

Note that if you select option **NLV** (No levels results) from the GEMPIE options menu, the prompts (as described above) proceed as if there were no levels results on the file. Thus option NLV affects the prompts and (if you choose to print totals rather than subtotals) what results you see.

The set of cumulatively-retained endogenous variables you select to print (from the choice menu presented just after you indicate the name of the Solution file) controls the endogenous variables for all types of results selected.

For example, if you choose to see totals and subtotals results, you will see these results for this selected set of endogenous variables.

If you choose to see totals and levels results, you will see these results for this set of endogenous variables (although, as usual, levels results may only be available for some of these selected endogenous variables).

It is not possible, for example, to ask GEMPIE in one run to show the totals results for one set of endogenous and to show the subtotals results for a different set of endogenous variables.

[17] If you selected option NLV (No levels results) from the GEMPIE options menu, case (2) applies instead.

7.2.2 Solution Files Produced by SAGEM

Recall from section 8.2 of GPD-3 that Solution files produced by SAGEM never contain levels results but may contain the totals results, subtotals results and/or individual column results.

In any one run of GEMPIE, you can do one of the following.

1) Print[18] any subtotals solutions (and the cumulative solution) stored on the Solution file.
2) Print any individual column results stored (by SAGEM) on the Solution file.
3) Setup and print new subtotals calculated from the individual column results stored (by SAGEM) on the Solution file.

If the Solution file you access via GEMPIE has both individual column and totals/subtotals results, you will be presented with the following choice selection.

```
CHOICE     MEANING
  t      Print TOTALS/SUBTOTALS already stored on the Solution file
  i      Print INDIVIDUAL COLUMN solutions already on the Solution file
  s      SETUP and print new SUBTOTALS from the individual column
           solutions already on the Solution file
```

The choices **t , i , s** correspond respectively to (1),(2),(3) above. If the Solution file you are accessing only contains individual column results, choice **t** will be omitted from the above menu. If the Solution file only contains totals/subtotals results, GEMPIE will tell you this and the menu above will not appear.

1) If you choose to print subtotals already stored on the Solution file, you will be asked which ones you wish to print. If you only want some of the subtotals stored to be printed, you indicate which ones by number (for example, 1-3, 5 to indicate 4 of the 6 available subtotals solutions). The subtotals you select will be printed in columns side-by-side and the cumulative solution will be printed as the last column.
2) If you choose to print individual column results, you will be asked to specify the columns (which are a subset of the individually-retained exogenous variables) and the rows (which are a subset of the individually-retained endogenous variables) to print.
3) If you choose to setup and print new subtotals, you will first be asked for the endogenous variables to be printed in each subtotal. (These are a subset of the individually-retained endogenous variables.) Then you specify the subtotals, one at a time, following essentially the same method as that used for creating subtotals when running SAGEM (see section 10.2 of GPD-3). Note that, when running GEMPIE, the subtotals are summed over some or all of the individually-retained exogenous variables (the only ones available on the Solution file) whereas in SAGEM they are summed over some or all of the shocked variables. Thus, in setting up one subtotal when running GEMPIE, you are choosing a subset of the individually-retained exogenous variables.
   Note that subtotals solutions you set up and print here are not stored on the Solution file by GEMPIE (which never alters a Solution file).

[18] When we say "print" in this section, we actually mean "put on the GEMPIE Print file". Once GEMPIE has written the solutions you desire on this Print file, you can choose whether to actually print this file (or view it in an editor etc).

7.3 Choosing Sets of Variables Interactively When Running GEMPIE

When running GEMPIE, you will sometimes need to make choices using the methods described here (for example, if you wish to choose the cumulatively-retained endogenous variables whose results appear on the GEMPIE Print file).

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.[19] (In the menu below, the words "cumulatively-retained endogenous" and "printed" will be replaced, more generally, by words describing the big set and the subset being chosen.)

```
     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.
   n     NO cumulatively-retained endogenous
            variables to be printed.
   m     All cumulatively-retained endogenous MACRO
            variables to be printed.
   1     All components of ONE cumulatively-retained ndogenous
          variables 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.)
```

GEMPIE Menu for Selecting Endogenous Variables to Print

The menu above is the one presented by GEMPIE if you are printing the totals results and you are selecting the endogenous variables to be on the GEMPIE Print file

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 are described in section 17.1.

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. Procedures for making such choices are described in section 17.3. In particular, two complete examples showing the selection of sets via the Lists option (‘L’) are given in section 17.3.

7.4 General Points About Output from GEMPIE

GEMPIE distinguishes between results which are identically zero and those which have been rounded to zero; it puts an asterisk '*' after the former, as in "0.0000*". For example, a result of 0.000023 will be rounded to zero if GEMPIE is reporting results to 4 decimal places and will be shown as "0.0000" (no asterisk). An example is in Table 2.7 in GPD-1, in which the result for p_PC("s1") is exactly zero while the result for p_XF("capital","s1") is only rounded to zero.

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

7.5 GEMPIE Options

The following two options let you choose the page width and page length of the GEMPIE Print file.

CPW Change page width
CPL Change page length

When GEMPIE prints out simulation results, the order of the rows (that is, of the endogenous variables) is determined by the order in which the VARIABLEs were declared the TABLO Input file. If you wish to print out the variables in some other order, select

RPO Choose row print order

Then, when you choose the endogenous variables to print, use the method 'f'(few) or 'L'(lists). The rows of the solution will be written on the Print file in the order you choose the variables.

By default, GEMPIE indicates identical solutions and shocks by messages saying they are "the same" as the previous ones. You can suppress these by choosing options

OSR Omit "the same" messages in results
OSS Omit "the same" messages in shocks

By default, if you are only printing a single solution (for example, the cumulative solution from a multi-step simulation), GEMPIE puts several components of the same variable on the same line (that is, across the page). You can suppress this by choosing option

SNA Single solution not across the page

Then the results for each component will be on a separate line. SNA also suppresses any Levels results if present.

By default GEMPIE puts blank lines in certain places to create "white space" in the Print file. You can suppress some of these by selecting either or both of the options

NBV No blank line before each variable
FBS Few blank lines for single solution across the page

If your Solution file contains Levels results, if you are printing the cumulative results, the levels results (Pre-simulation, Post-Simulation and Change) are shown by default. Select option

NLV No Levels results

levels results will not be shown. See section 7.2.1 for more details.

CHAPTER 8

8. SLTOHT: Processing Simulation Results

8.1 Solution File to Header Array/Text/Spreadsheet file

The program SLTOHT (**S**o**L**ution **TO H**eader or **T**ext) can be used to convert any solutions on a Solution file to

- a Header Array file, ready for viewing in ViewHAR or via another Header Array file program,
- to a GEMPACK text file in one of several formats or
- spreadsheet output such as a CSV file[20] which makes it easy to import the solution values into a spreadsheet program[21].

The idea of SLTOHT is to make the solution values available for processing or reporting (for example, for producing tables and graphs).

SLTOHT breaks up the solutions into several **arrays**, one corresponding to each variable in the model.

In sections 8.2 and 8.3 below, we introduce you in more detail to the different sorts of output that SLTOHT can produce. The remaining sections of this chapter give more detail about Header Array and GEMPACK text file output from SLTOHT. Spreadsheet output from SLTOHT is discussed in more detail in chapter 9.

8.2 SLTOHT Examples

The following examples illustrate some of the ways SLTOHT can be used to manipulate results on the Solution file and put them in the format that you may want to view or use in a report. Here we use the program WinGEM, but SLTOHT can be run interactively from the Command prompt; you could carry out the following examples by selecting suitable initial options and responding to prompts.

8.2.1 Totals Solution to Header Array File

For this example, you will work with the Solution file SJLB.SL4 created by running GEMSIM or the TABLO-generated program SJ.EXE with the Command file SJLB.CMF. (This is the Solution file from the example simulation described in section 2.5 or 2.6 of GPD-1.)

Run WinGEM, set your working directory to the directory for Stylized Johansen (C:\SJ).

To run SLTOHT, select from WinGEM's menu

OtherTasks | Solution file to Header/Text (SLTOHT)

In the SLTOHT Window which appears, ***Select*** the Solution file SJLB.SL4. A box will appear asking you to choose the type of solutions to print with a choice of *Totals* or *Totals and levels* .

---

[20] CSV stands for Comma Separated Values. The comma acts as a separator between spreadsheet cells.

[21] We are grateful to Kevin Hanslow for suggesting spreadsheet output and for providing us with a prototype implementation of this.

For the first example below choose ***Totals*** and then select the ***Run*** button.

View the file created by SLTOHT called SJLB.SOL. This is a Header Array file which contains Headers called *0001, 0002, 0003,….* Each Header Array contains the simulation results for a single variable.

ViewHAR Contents Screen

```
      Header   Type Size          Coeff[22]
1     0001     RE   1             p_Y    ..
2     0002     RE   SECT          p_PC   ..
3     0003     RE   FAC           p_PF
4     0004     RE   SECT          p_XCOM
5     0005     RE   FAC           p_XFAC
and so on
```

For example header *0001* contains the results for variable *p_Y*, header *0002* for variable *p_PC*.

This is the simplest way of running SLTOHT. If you were running SLTOHT interactively at the Command prompt, this example is equivalent to running SLTOHT with no options selected.

With no options selected, SLTOHT just converts the Solution file to a Header Array file.

8.2.2 Totals Solution to CSV File for Spreadsheet (SSE)

If you want to make a text file or a CSV file for spreadsheets, you must select some Options.

Return to the SLTOHT window in WinGEM. In the SLTOHT menu, select

Options | SLTOHT Options

and click on the option[23]

- ***SSE*** ***Spreadsheet output with element labels***

Select the Ok button to return to the SLTOHT window and ***Run*** the program.

This time SLTOHT will make the file SJLB.CSV which you can ***View***.

After some comments at the top of the file (lines starting with !), SLTOHT prints out all the variables on the Solution file. This is in Comma Separated Value (CSV) style since all the numbers are followed by commas.

```
! Variable p_Y # Total nominal household expenditure #
 ! Variable p_Y of size 1
   5.8852696    ,

 ! Variable p_PC # Price of commodity i #
 ! Variable p_PC(SECT) of size 2
 s1          ,  0.0000000    ,
 s2          ,-0.94857663    ,

! Variable p_XF # Factor inputs to industry j #
 ! Variable p_XF(FAC:SECT) of size 2x2

             ,s1             ,s2             ,
 labor       ,  9.9999990    ,  10.000002    ,
 capital     ,-4.48017488E-07, 4.48017488E-07,

and so on till all variables have been output.
```

[22] Because ViewHAR is usually used for data files containing data to be read into Coefficients, Coeff is used in this heading for the column containing variable names.

[23] If you were running SLTOHT interactively at the Command prompt, this example is equivalent to running SLTOHT with option SSE selected.

Note that a two dimensional variable such as p_XF with indices ranging over the sets FAC and SECT, is output as a matrix with Element labels for the rows and columns. Set SECT in the Stylized Johansen model contains the two elements *s1, s2* while set FAC contains the two elements *labor* and *capital*. If you import the file SJLB.CSV into a spreadsheet program (for example, Excel), the results for p_XF (and other two-dimensional variables) are seen in a table suitable for use in reports.

Table of p_XF(FAC,SECT) results

	s1	s2
labor	9.9999	10.0000
capital	0.0000	0.0000

This is the intention behind option SSE, see chapter 9.for more details.

8.2.3 Cumulative + Levels Solution to Header Array File

Now we suggest that you again select the file SJLB.SL4 in the SLTOHT window.
This time choose[24] *Totals and Levels* instead of *Totals.*

Return to the SLTOHT window. First select ***Options | SLTOHT Options***

and select ***Output Header Array file*** (instead of SSE.) Select ***Ok***.

Click on the ***Solutions*** button and select ***Totals and levels solutions*** and then ***Ok***

Click on the ***Run*** button to run SLTOHT. When asked[25], change the name of the file produced to SJLBLEV.SOL. ***View*** the file produced.

	Header	Type	Size	Coeff
1	0001	RE	Lin+Lev	p_Y
2	0002	RE	SECT*Lin+Lev	p_PC
3	0003	RE	FAC*Lin+Lev	p_PF
4	0004	RE	SECT*Lin+Lev	p_XCOM
5	0005	RE	FAC*Lin+Lev	p_XFAC
and so on				

Each of the separate Headers shows the cumulative[26] results (Linear) plus the levels solutions (PreLevel, PostLevel, Change). The set *Lin+Lev* contains the elements (Linear, Prelevel, PostLevel, Change). The file produced contains

- a 1 x 4 array for the variable *p_Y* showing the value of its only component in each of these 4 solutions (see Header *0001*)

	Linear	PreLevel	PostLevel	Changes
1	1.000	6.000	6.060	0.060

[24] If you were running SLTOHT interactively at the Command prompt, this choice is equivalent to running SLTOHT with option SHL selected.

[25] SLTOHT makes up a default name based on the name of the Solution file. For example for Header Array output for Solution file SJLB.SL4, the default name is SJLB.SOL. However we have already used this name in the example in section 8.2.1 so change the name to SJLBLEV.SOL.

[26] The cumulative results are sometimes referred to as the totals results (as, for example, in the WinGEM form which asks you which solutions you wish to work with).

- a 2 x 4 array for the variable *p_PC* showing the values of its 2 components in these 4 different solutions (see Header *0002)*

```
          Linear        PreLevel      PostLevel     Changes
s1        0             1.000         1.000         0
s2        -0.949        1.000         0.991         -0.009
```

and so on.

8.2.4 Subtotals + Cumulative Solution to Header Array File

In section 11.3 of GPD-3, there is an example of running GEMSIM or the TABLO-generated program SJ.EXE with a Command file SJSUB.CMF to obtain some subtotal results on the Solution file SJSUB.SL4.

This Solution file SJSUB.SL4 from a multi-step simulations with Stylized Johansen contains
1 cumulative solution,
3 levels solutions (PreSimulation, PostSimulation and Change),
2 subtotal solutions,
making 6 solutions in all.

SLTOHT will allow you to show either

- the totals and the levels solutions, or
- the totals and the subtotals solutions, or
- just the totals solution.

In the SLTOHT window, ***Select*** the file SJSUB.SL4.

Choose the *Totals and subtotals solutions* and then ***Run*** SLTOHT to make Header Array output as in the previous example.

Each of the separate Headers show two subtotal results and the cumulative results. For example,

- a 1 x 3 array for the variable *p_Y* showing the value of its only component in each of these 3 results, (see Header *0001*)

```
   SubTot1       SubTot2       Cumulative
1  6.111         7.785         13.896
```

- a 2 x 3 array for the variable *p_PC* showing the values of its 2 components in these 3 different results,

```
   SubTot1       SubTot2       Cumulative
s1        0             0      0
s2  -0.957          1.831      0.874
```

and so on.

The set SOLUTIONS contains the elements (SubTot1, SubTot2, Cumulative).

8.3 Mapping Files

There are two type of Mapping Files used by SLTOHT, one for Header Array output and one for spreadsheet output. Both types are used to choose the variables on the Solution file that you wish to write to the output file. Variables are output in the order that they are listed on the Mapping file.

A **Header Array mapping file** is basically a list of variables on the Solution file to be output with an associated Header for each variable. Details of syntax are given in section 8.6.1.

A **Spreadsheet Mapping file** is a list of variables on the Solution file that you wish to output. You can also specify individual components - see section 9.1. Spreadsheet Mapping files can be used with the SLTOHT options SS and SSE, and must be used with options SSS and SHK.

For example, if you only wish to see the results for variables *p_Y, p_PC* and *p_XF,* the Spreadsheet Mapping file would be a text file created in a text editor containing just the following three lines. (Don't leave any spaces at the beginning of the lines.)

Spreadsheet Mapping File

```
p_Y
p_PC
p_XF
```

Create this file in the editor and save it as the file SJSS.MAP.

8.3.1 Example of a Spreadsheet Mapping file (Option SSS)

In WinGEM, in the SLTOHT window, select

Options | Use Mapping File

and give the name of the Spreadsheet Mapping file SJSS.MAP.

To select the SLTOHT options for spreadsheets, select

Options | SLTOHT Options

and select option **SSS Short spreadsheet output** and choose a Comma as the separator (this is the default).

Select the Solution file you want to look at, for example SJLB.SL4 and select just *Totals solutions* .

Run SLTOHT and change the name to SJLBSSS.CSV. ***View*** the output file.

SSS Output using Mapping file

```
Solution,d:\sj\sjlb,
 p_Y,  5.8852696    ,
 p_PC(s1),  0.0000000    ,
 p_PC(s2),-0.94857663    ,
 p_XF(labor:s1),  9.9999990    ,
 p_XF(capital:s1),-4.48017488E-07,
 p_XF(labor:s2),  10.000002    ,
 p_XF(capital:s2), 4.48017488E-07,
```

Apart from the first line which gives the name of the Solution file, each line gives the name of the variable (including the element names) followed by the value of the variable.

Option SSS leaves out all the usual comments at the start of the file. It is useful for creating simple tables for reports, just picking out interesting variables.

If you open the file SJLBSSS.CSV in Excel or some other spreadsheet program, it can be quickly made into a table as shown below.

Spreadsheet Table

Solution	D:\sj\sjlb
p_Y	5.89
p_PC(s1)	0.00
p_PC(s2)	-0.95
p_XF(labor:s1)	10.00
p_XF(capital:s1)	0.00
p_XF(labor:s2)	10.00
p_XF(capital:s2)	0.00

8.3.2 Example of Matrix Variables for Spreadsheets (Option SSE)

In section 8.2.2, you ran SLTOHT without using a Mapping file and the output was all the variables on the file. If you rerun this example, but this time using the Mapping file SJSS.MAP, you will get just the three variables shown in the results table in section 8.2.2 and no other variables.

8.4 Running SLTOHT Interactively

[You can omit this section if you always use WinGEM.]

If you need to carry out some unusual task, you may wish to run SLTOHT interactively to give you full control of the program. Different output is produced mainly by choosing different options at the start of SLTOHT. Each of the initial options for SLTOHT has a short help screen available by typing in a question mark ? followed by the three letter code for the option, for example ?SSE.

8.4.1 Choosing Solutions for Output

After you have chosen the initial options, SLTOHT prompts for the Solution file name, reports the contents of the Solution file to the screen and presents a menu of possible solutions to be written to arrays. The range of choices depends on what SLTOHT finds on the Solution file. The types of solutions which can be on a Solution file depends on whether the Solution file was produced by SAGEM or by GEMSIM or a TABLO-generated program.

Solution files produced by GEMSIM or TABLO-generated programs

A Solution file produced by GEMSIM or a TABLO-generated program always contains the cumulative (that is, totals) solution and may contain levels results and/or subtotals results (see section 8.2 of GPD-3). On any one run of SLTOHT, you cannot see both the levels results and subtotals results. If you want to see levels results you must select option **SHL** (see section 8.5 for details). If you want to see subtotals results (see chapter 11 in GPD-3), you indicate this by responding appropriately to the prompts. SLTOHT does not let you choose to see only some of the subtotals results – you either see all or none.

If you output the cumulative and subtotals solutions, the subtotals results come before the cumulative results in the final dimension of the arrays output by SLTOHT.

If you output the cumulative and levels solutions, order of solutions is

1. the linear solution (the percentage change or ordinary change results)
2. pre simulation levels value
3. post-simulation levels value
4. change from pre simulation levels to post simulation

For example, if the Solution file contains only individual solutions[27], then the only choice is to write all these solutions, or exit. If the Solution file contains individual, subtotal and cumulative solutions, then the choice is to

1. write all of these solutions,
2. write just the totals (that is, subtotals and cumulative solution),
3. write just the cumulative solution.

The solutions represented by the final dimension in an array have the order: the individual solutions (if chosen), then any subtotals chosen, then the cumulative solution (if present on the Solution file).

8.4.2 Shocks and Undefined Solution Values

In the solutions shown by SLTOHT, for shocked components of variables, the appropriate shock value (for that solution) is given as the value of the component of the variable in the output array.

The solution for any exogenous variable, none of whose components were shocked, is an array of zeros.

Some variables may be undefined for a solution if not all endogenous variables were retained on the Solution file (see, for example, the "cumulatively-retained endogenous" statement described in section 8.1 of GPD-3). Any component which is undefined for a solution has a very big number entered in its position in the array. This number is currently set at 10**10 (that is, 10 to the power 10). When SLTOHT writes the arrays to the output file, it reports at the screen for each array written (i) the variable name, (ii) the header name, (iii) the size of the array and (iv) the number of such undefined values found.

8.5 Levels Results – Options NLV and SHL

This section applies to Solution files produced by GEMSIM or TABLO-generated programs.

Option **NLV** "Report no levels values" is the default in SLTOHT.[28]

This means that, if the Solution file also contains levels solutions, you can not see these solutions unless you choose the option **SHL** or choose the *Totals and levels solutions* in WinGEM as in section 8.2.3. If you have chosen to output levels values, you cannot see subtotal solutions in the same run.

If you output levels solutions, see section 8.4.1 for the order in which the results appear.

You can choose to report just one of the levels results. Consult the on-line help by entering ?LV at the initial option screen to find out about the options LVB, LVA, LVC and NLV.

[27] Individual solutions are produced only by SAGEM (see chapter 10 of GPD-3).

[28] This information was omitted from the Release 6.0 documentation.

8.6 Header Array file Output

For Header Array file output, SLTOHT writes a Header Array of solution values for each variable in the model. The solutions are selected from solutions on the Solution file.

> For example, for a variable with 3 arguments, of size (3 x 2 x 4), if you select t solutions from the Solution file, then the Header array written is an 4-dimensional array of size (3 x 2 x 4 x t).

The last (that is, the fourth) dimension represents the different solutions you choose in the order

1. any individual solutions (only possible if SAGEM created the Solution file),
2. any subtotal solutions,
3. the cumulative solution,
4. any levels results (only possible if the Solution file was created by GEMSIM or a TABLO-generated program).

Each array (or variable) is associated with a different header on the Header Array file written. You can specify the headers to be used, using a Header Mapping file as described below. If you do not supply such a Mapping file, SLTOHT chooses the headers for you, namely "0001" for variable number 1, "0002" for the next one and so on.

8.6.1 Header Mapping file

A Header Mapping file can be used for cases which produce Header Array file output. It is a text file, which maps 4-letter headers to variable names, and also optionally provides some details about the number and size of the dimensions of the variable.

If you supply a Header Mapping file containing headers for only some of the variables, only solutions for these variables will be put on the Header Array file produced by SLTOHT.

We recommend that you use a Header Mapping file if

- you want results for only some of the variables to appear in the Header Array file produced.
- you want to assign meaningful headers to the different variables. If you are using the Header Array file produced by SLTOHT as an input to another program (perhaps a TABLO-generated one), you should certainly do this to ensure reliable communication between the two programs.[29]

Each line of a Header Mapping file must have the following format:

<Header> <Variable Name> [<No of Dims> <Dims>]

where <Header> is a 4-letter Header (must be in columns 1-4)
<Variable Name> is up to 15 characters long (must begin in column 6)
<No of Dims> is the number of dimensions for this variable
<Dims> is a list of the number of components in each dimension

For example, using the default headers 0001, 0002,... the Header Mapping file for Stylized Johansen solutions is as follows:

[29] The association between the “0001”, “0002” etc headers and variables if you do not specify a Header Mapping file will change if you subsequently add variables to the model. This could destroy communication if you use “0001” and so on in your second program. This problem disappears if you use a Header Mapping file.

Sample Header Mapping file for Stylized Johansen

```
0001 p_Y                  1      1
0002 p_PC                 1      2
0003 p_PF                 1      2
0004 p_XCOM               1      2
0005 p_XFAC               1      2
0008 p_XF                 2      2    2
```

The number of dimensions <No of Dims> and the list of dimensions <Dims> are optional and are usually omitted, as in the example below.

Alternative Sample Header Mapping file

```
DVFI p_DVFACIN
DVHS p_DVHOUS
XFAC p_XFAC
Y    p_Y
```

In the example above, we have chosen meaningful headers (abbreviations for the variable names). These are usually more reliable than the 0001, 0002 (and so on) chosen by SLTOHT. In particular, you do not need to change this mapping file if you subsequently add or delete other variables from the model.

Note that the first 4 characters in each line are for the header (leave spaces as appropriate as in the p_Y example above) and that the variable name must start in column 6.

SLTOHT checks the syntax of the Header Mapping file when it is first opened and reports comprehensively on any syntax errors. SLTOHT will also check whether there are any inconsistencies between the details on the Mapping file and the actual solutions found on the Solution file. Usually the Solution file contains information about the dimensions of variables and set sizes involved, and this information is taken into account.

If a Header Mapping file is not chosen, then SLTOHT will create one using a default set of Headers for the variables and using whatever dimensioning information is available on the Solution file to determine the variable dimensions. You can then edit this mapping file created by SLTOHT and use it for subsequent runs of SLTOHT. For example, you can change the headers associated with each vector variable to ones more meaningful than the "0001" and so on used by SLTOHT as a default.

8.6.2 Alternative Header Array File Output - Variable Arguments Ignored (VAI)

This alternative form of Header Array file output ignores the arguments of variables, treating each variable as if it were a one-dimensional variable. To obtain this output, select option

VAI Variable arguments ignored (har output)

in the option choice at the start of SLTOHT.

In the example above the 3 x 2 x 4 variable would be considered as if it were a one-dimensional variable with 24 components.

For a variable z(i,j,k), the order of variable elements is with the first index changing most rapidly (see section 5.3 of GPD-3). So the 24 components come in the order

z(1,1,1), z(2,1,1), z(3,1,1), z(1,2,1), z(2,2,1)....

[For example, this is the same order as on a GEMPIE Print file.]

As before, you can choose different solutions from the Solution file and output the different solution values as the columns of a 2-dimensional Header array for each variable.

In the example above, if t solutions are chosen and option 'VAI' is selected, the Header array for this variable z is of size (24 x t).

8.7 GEMPACK Text File Options (SIR,SIC), including Spreadsheet (SS)

The SLTOHT options SIR, SIC and SS all produce text file output. Since the files produced are just text files, they can be read or edited in any text editor.

The text files produced are in a special format used in GEMPACK which we refer to as "GEMPACK Text file" or as a “GEMPACK Text Data file”. If you need to read your SLTOHT output file back into a GEMPACK program such as MODHAR, you need to use a GEMPACK text file. GEMPACK programs are able to read text data files if they are written in the form described in chapter 6 "Syntax of GEMPACK Text Data Files".

To produce a GEMPACK text data file, select from the options menu, presented when SLTOHT starts, one of the options SIR, SIC or SS (without a Spreadsheet Mapping file):

SIR Solutions in rows (text output)
SIC Solutions in columns (text output)
SS Spreadsheet (Sols in cols - text)

Note that option SS only produced a GEMPACK text data file if you do not specify a Spreadsheet Mapping file (see section 9.1)[30]

SIC and **SS** produce as output arrays for each variable in which

- each column specifies a different solution, and
- the rows specify the components in turn for a given variable.

  In the example above of a (3 x 2 x 4) variable, and t different solutions, this will result in output on the text file of an array with 24 rows by t columns for this variable. The variable arguments are ignored (as in option VAI for Header arrays), effectively collapsing the first 3 dimensions into a single dimension to allow the columns to represent different solutions.

SIR (solutions in rows) is similar to **SIC** except that

- each row specifies a different solution, and
- columns specify different components of the variable.

  This, for the example above, gives an array of t rows and 24 columns.

In each case for text file output, "how much data" information is output before the data values of the array. Details of the "how much data" information and structure of text files are described in chapter 6.

SIR and **SIC** outputs have only four or five (real) values per line. If the array to be output has more values than this, the entries are continued on to the next line until the end of the row is reached. Each row starts on a new line.

Option **SS** allows longer lines, continued to the maximum record length, and is suitable for input to a spreadsheet. The separator between data values can be chosen to be a comma or some other character such as a TAB character.

Below is a section of output from SLTOHT for the variables p_XH and p_XC of Stylized Johansen with the option **SS** (spreadsheet) chosen. 'SIC' would give similar output. Comments following the exclamation mark ! are output by SLTOHT as a guide to the form of output.

[30] Each line begins with the name of the variable and its arguments – see the example output file shown in section 9.1.

Sample of spreadsheet text output (SS, no mapping file)

```
! For each variable below columns represent the following solutions:
! 2 INDIVIDUAL column(s), each one giving the solution
!  for an individual shock.
! 1 SUB-TOTALS column(s) giving cumulative solutions
!  across sub-sets of the complete set of shocks.
! 1 TOTALS column, giving cumulative solutions across ALL shocks.
! For details about the shocks relevant to each column, run GEMPIE.

! ************************   SOLUTIONS   ************************

! Variable p_XH # Household demand for commodity i #

! Showing this as an array of size 2x4

! Solutions in columns, each row is a different component

2,4, real spreadsheet ;

     0.600000    ,   0.400000    ,    1.00000    ,    1.00000    ,
     0.700000    ,   0.300000    ,    1.00000    ,    1.00000    ,

! Variable p_XC # Intermediate inputs of commodity i to industry j #

! Showing this as an array of size 4x4

! Solutions in columns, each row is a different component

4,4, real spreadsheet ;

     0.600000    ,   0.400000    ,    1.00000    ,    1.00000    ,
     0.700000    ,   0.300000    ,    1.00000    ,    1.00000    ,
     0.600000    ,   0.400000    ,    1.00000    ,    1.00000    ,
     0.700000    ,   0.300000    ,    1.00000    ,    1.00000    ,
```

SS output optionally allows you to use a Spreadsheet Mapping file to specify exactly which variables and which components you wish to output as described in section 9.1. However if you use a Spreadsheet Mapping file with option SS the text file produced is not in GEMPACK Text Data File format as described in chapter 6 and so cannot be read by other GEMPACK programs (unless you carry out some further editing).

SLTOHT has 2 other options for spreadsheet output - **SSE** and **SSS** (see chapter 9). Although these two options create a text file, it is not a GEMPACK text data file suitable for reading into GEMPACK programs using the syntax defined in chapter 6.[31]

See chapter 9 for more details on spreadsheet options SS, SSE and SSS.

[31] Each line begins with the name of the variable and its arguments – see the example output file shown in section 9.1.

8.8 Shock CMF Output - Option SHK

This is an option developed for use with the recursive dynamic models such as MONASH (see section 10.1). When carrying out policy simulations (deviations from a base forecast) with such models, 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.[32] The statements produced using option SHK are of the form

```
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 – see section 12.2.)

The program RunDynam (see section 2.7) uses option SHK when preparing for the base re-run and the policy run.

8.9 Other SLTOHT Options

Spreadsheet options SS, SSS and SSE are covered in chapter 9 "SLTOHT for Spreadsheets".

On-line help is available on all SLTOHT options. Enter a question mark ? followed by the option to see a brief help screen, for example ?NEL to see help on option NEL.

The option NEL is used with option SIR and SIC. Usually on SIR and SIC output the component numbers and solution numbers are written as comments as part of each array. If you choose

NEL No element labels for SIR, SIC, SS, SSS

these numbers will be omitted. With SS and no mapping file, NEL has no effect. With SS or SSS and a mapping file, NEL replaces the element names with component numbers, for example p_XF(1) instead of p_XF(labor:s1).

To output the results separately, consult the on-line help for

SEP Show solutions separately

If you choose option SEP,

- each column of results is output to a separate file if you are producing Header Array output.

[32] Previously to prepare shocks for such a policy 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 (written by mark Horridge) to convert the CSV output to the desired shock statements.

- the different columns of results will be shown all on the same output file, one after another, if you are producing SS or SSE output.

The option DES is rarely used. You can choose to produce a Description file, by choosing the option

DES Write description file

at the start of SLTOHT. This is a file which contains the following information, in the stated order :

(a) The verbal description of the simulation as entered originally when you ran SAGEM, GEMSIM or the TABLO-generated program to create the Solution file. The description is terminated by a line beginning with '*'.

(b) If individual solutions are chosen, for each individual solution, the name and component number of the shock which gave rise to that solution. Each such individual description is terminated by a line beginning with '*'. The component number is reported as if the shocked variable were one dimensional.

(c) If subtotal solutions are chosen, for each subtotal solution, the description of that subtotal, as entered by the user in SAGEM, terminated by a line beginning with '%'. Then a list of the names (one name per line) of those variables of which there is at least one shocked component that contributed to this subtotal. Each subtotal description is terminated by a line beginning with '*'.

(d) If the cumulative solution is on the Solution file, a list of the names (one name per line) of the variables of which there is at least one shocked component. The cumulative totals description is terminated by a line beginning with '*'.

CHAPTER 9

9. SLTOHT for Spreadsheets

One of the main uses of the GEMPACK program SLTOHT is to convert the simulation results on a Solution file into text files suitable for input into spreadsheet programs such as Excel. Often these text files are Comma Separated Values (CSV) files where the comma acts as a separator. However other separators can be used, for example, a space or a tab character.

Examples of the way SLTOHT can be used to produce spreadsheet files have been given in sections 8.2.2 and 8.3. This chapter contains complete details about producing spreadsheet files using SLTOHT.

In most cases, SLTOHT is run using a Spreadsheet Mapping file to select the variables you wish to write to the output file. The use of Spreadsheet Mapping files (introduced in section 8.3 above) is described in section 9.1 and their syntax is given in section 0.

The spreadsheet options are

SS Spreadsheet (Sols in cols)

SSS Short SS output

SSE Spreadsheet with element labels

The main difference between SS and SSS output is that SS output contains comments and other information on the top of the file whereas SSS contains no comments, just the variables you asked for in the Spreadsheet Mapping file. SSE is very useful when you have just one solution, for example the cumulative solution, and two dimensional variables. In this case a table is produced, with the rows and columns labelled with element names. For example in section 8.2.2, the variable p_XF is output with rows (labor,capital) and columns (s1, s2).

The syntax of Spreadsheet Mapping files allows you to specify subsets and elements of variables - see section 9.1 and 9.3.

SLTOHT can also be used to produce tables of variables in which results **for different variables** are written side-by-side in the columns - see section 9.4.

9.1 Spreadsheet Mapping Files

It is often useful to choose variables and components in a flexible way. In section 8.3 mapping files, both Spreadsheet and Header Array mapping files, were introduced and an example of the use of a Spreadsheet Mapping file was given in section 8.3.1.

If you are using WinGEM, you need to choose the spreadsheet option by selecting

OtherTasks | Solution to Header/Text (SLTOHT) | Options | SLTOHT Options

and then choose the option you want to use SS, SSS or SSE. You also need to choose ***Use mapping file*** and choose its name.

[If you are running SLTOHT interactively, select the spreadsheet option you need at the start and later in the prompts, you will be asked whether you want to use an existing Spreadsheet Mapping file.]

The Spreadsheet Mapping file specifies the variables to be output and their components as shown in the examples below:

- **Simple Spreadsheet Mapping File**

The example in section 8.3.1 is a simple list of variable names (all components of each).

```
p_Y
p_PC
p_XF
```

- **Using SEENV to make a Spreadsheet Mapping file**

You can create a Spreadsheet Mapping file from a Solution file using the program SEENV. You can use this method to avoid the chore of typing in all the variable names from your model. Details are given in section 12.2.

- **Subsets and Elements in Mapping file**

An example of a Spreadsheet Mapping file specifying elements and subsets[33] is the following:

```
p_XF(FAC,"s1")
p_XC(NUM_SECT,SECT)
```

See section 9.3 for another example of this with option SSE.

- **Side-by-side variables in Mapping file**

The line below will put the p_PC and p_XH results side by side in the same table.

```
p_PC : p_XH
```

See section 9.4 for examples with SS or SSS where several variables are side by side.

- **Numerical Components in Mapping file**

An example where the components are listed numerically is given below:

```
p_xH
p_XC
  2-3 1 1 1 ;
p_PF
  1
  2 ;
```

For a case in which four solutions were requested, this results in the output as shown below (except that, below, the comment lines relating to variables p_XC and p_PF have been omitted).

[33] This syntax was introduced for Release 6.0 of GEMPACK.

Sample of SS spreadsheet output using Spreadsheet Mapping file

```
! Variable p_XH # Household demand for commodity i #
! Showing this as an array of size 2x4
! Solutions in columns, each row is a different component.
! First column must be removed before using array as GEMPACK input
2 , 4 , real spreadsheet ;
p_XH(s1),   0.600000   ,   0.400000    ,   1.00000    ,   1.00000    ,
p_XH(s2),   0.700000   ,   0.300000    ,   1.00000    ,   1.00000    ,


5 , 4 , real spreadsheet ;
p_XC(s2:s1),   0.700000   ,   0.300000    ,   1.00000    ,   1.00000    ,
p_XC(s1:s2),   0.600000   ,   0.400000    ,   1.00000    ,   1.00000    ,
p_XC(s1:s1),   0.600000   ,   0.400000    ,   1.00000    ,   1.00000    ,
p_XC(s1:s1),   0.600000   ,   0.400000    ,   1.00000    ,   1.00000    ,
p_XC(s1:s1),   0.600000   ,   0.400000    ,   1.00000    ,   1.00000    ,
2 , 4 , real spreadsheet ;
p_PF(labor),  -0.400000   ,   0.400000    ,  0.000000E+00,  0.000000E+00,
p_PF(capital),   0.600000   ,  -0.600000    ,  0.000000E+00,  0.000000E+00,
```

9.1.1 Syntax Rules for Spreadsheet Mapping Files

The rules for a Spreadsheet Mapping file are as follows.

- Variable names must start in the first position in a line.
- Variable names can be followed by sets, subsets or elements in brackets (on the same line) to specify components
  or
  Components can be listed in one or more lines immediately following the variable name. All such lines must be indented at least one space to the right and the final line terminated with a semi-colon.
- Variables with no components specified will have all components output.
- Variables not specified will not appear on the output.
- Components may be repeated and may be in any order.
- By default, element names are used in output when you are using a Spreadsheet Mapping file.
- If you are using element numbers instead of element names (option NEL with SS or SSS), multidimensional arrays are treated as linear; thus p_XC(3) would refer to p_XC("s1","s2").
- Fatal errors occur if you don't indent, specify a component outside the range or leave out the semi-colon.
- Variables can be specified separated by colons : to be output side-by-side (subject to the restrictions in section 9.4).
- With option SSE, % characters can be used before a set name to indicate how you want 3 and higher dimensions to be grouped. See section 9.3.2 for details.

9.2 Options Spreadsheet (SS) and Short Spreadsheet (SSS)

SS was the original spreadsheet option used to produce a CSV file. If used without a Spreadsheet Mapping file, SS produces a GEMPACK text data file in spreadsheet order as in the example in section 8.7.

If you run SLTOHT **with a Spreadsheet Mapping file and option SS**, solutions are shown as different columns (or as a single column if you have only one solution). Each row is a different component of a variable, in the order of variables listed on your Spreadsheet Mapping file. At the top of the file, there are various comments about the Solution file (lines which start with an exclamation mark !).

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 into a spreadsheet.

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 specify the variables in your output file.

9.2.1 Component Numbers rather than Element Names (SS or SSS)

This section applies when options SS or SSS are selected and you are using a Spreadsheet Mapping file.

By default, element names are used (when they are available) in the output, as in, for example,

```
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 instead of element names, you can see these if you select option "**NEL**" from the SLTOHT options menu. For example,

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

9.3 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.
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 percentage 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

You can specify subsets (and elements) on the Spreadsheet Mapping file if you are only interested in a (rectangular) part of a variable.[34] 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

[34] Option SSE was new for Release 5.2, but the use of subsets and elements in spreadsheet mapping files was new for Release 6.0.

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". [By default, 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. However you can alter this order – see section 9.3.2 below.]

9.3.1 Output of Levels Results Using Option SSE

If levels results are available (see section 8.2 in GPD-3), 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.

9.3.2 Three and Higher Dimensional Arrays of Results

This section considers output for option SSE for arrays with 3 or more dimensions.

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,%reg,%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.

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

It is possible to specify on a **Spreadsheet Mapping file** (see section 9.1) 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.[35]

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 in a 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.

---

[35] This was introduced in Release 6.0.

A more realistic example comes from the GTAP model with 3 tradeable commodities food,mnfcs,svces and 3 regions USA,EU,ROW (see the example at the beginning of section 9.3), 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.

9.5 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).

CHAPTER 10

10. ACCUM and DEVIA : Accumulation and Differences

Program ACCUM assembles the results of several different but related simulation runs into a spreadsheet file. For example, the related simulations could be different years of an economic forecast.

DEVIA produces a spreadsheet file which shows the differences between two sets of results, for example, between the policy (deviation) simulations and the base case simulations from a recursive dynamic model such as MONASH.

We describe these programs in detail in this chapter.[36]

ACCUM can produce accumulated results as well as side-by-side results. See the options ACC,ACI described in sections 10.2.2 and 10.2.3.

DEVIA can produce year-on-year differences. See the option NAC described in section 10.3.2.

DEVIA can be told the name of a Solution file associated with the results being processed - see the option SOL described in section 10.3.1. If you use this option, DEVIA will correctly process change and percentage-change variables.

The Release 7.0 versions of ACCUM and DEVIA run considerably more quickly than the original Release 6.0 versions.

On-line help on options is available by typing a question mark ? followed by the option code. For example, for the option ACC in ACCUM, type ?ACC.

10.1 Use with Dynamic Forecasting Models

Programs ACCUM and DEVIA are useful with recursive dynamic models (such as MONASH and GTEM) which are solved several times in a year-on-year fashion to produce a series of Solution files. A description of the MONASH model is given in Dixon and Rimmer (2000).

Dynamic models such as MONASH and GTEM 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.

[36] Programs ACCUM and DEVIA were based on Mark Horridge's programs ACCUMSOL and DEVIATE which only work on PCs since they are written in Pascal. We have rewritten and extended these 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. ACCUM and DEVIA were introduced in Release 6.0 of GEMPACK. We are grateful to Mark Horridge for allowing us to modify his programs.

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.

In particular ACCUM and DEVIA are used in conjunction with the Windows interface **RunDynam** (or is model-specific versions including **RunMONASH** and **RunGTEM**) described in section 2.7.[37]

10.1.1 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 9.2 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.

Table 10.1.1 : 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

10.2 ACCUM

By default, ACCUM simply puts the results from the different simulations side by side. This can be useful for any model, not just dynamic models.

10.2.1 Columns Showing Totals and Averages

If you wish, ACCUM will also calculate three extra columns which contain the arithmetic total of the preceding columns, the geometric total and the average annual growth rate. See, for example, the final three columns in Table **10.1.1** above.

When you run ACCUM, you are asked if you wish to have these extra columns added. You can respond simply "y" (yes) or "n" (no). However if you respond simply "y" ACCUM does not know

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

which variables represent percentage changes and which represent ordinary changes. Provided you still have available one of the Solution files which were used to produce the spreadsheets being put side by side, it is best to respond to this prompt with the name of the Solution file. Then ACCUM will take it that you want these extra columns and will be able to tell from the Solution file which variables are percentage changes and which are ordinary changes. This will be used in producing the last two of the extra columns, as explained below.

- If a variable is known to be a change variable, the last two columns show the arithmetic totals and arithmetic average of the results from the different simulations.
- If a variable is known to be a percentage-change variable, the last two columns show the geometric total and the geometric average of the results from the different simulations.

The formulas used to compute 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

10.2.2 Year-On-Year Results or Accumulated Results (Option ACC)

By default, ACCUM simply puts the results side by side. If these come from several years of the base case, or a policy simulation, with a dynamic model, you would interpret these as **year-on-year results** (that is, as changes or percentage changes over the year).

ACCUM can also produce **accumulated** results which show the change or percentage change relative to the value at the beginning of the first year of the base case or policy. You should specify option **ACC** when running ACCUM if you want these accumulated results. For example, if the year-on-year results for a change variable c1 and a percentage-change variable p1 are

	1998	**1999**	**2000**	**2001**
c1	22.0	13.0	14.0	26.0
p1	2.0	1.0	1.0	2.1

the accumulated results would be

	1998	**1999**	**2000**	**2001**
c1	22.0	35.0	49.0	75.0
p1	2.0	3.02	4.05	6.235

The change results are added and the percentage-change results are compounded. For example, the 49.0 value under the year 2000 means that this is the total change in c1 from its value before the start of year 1998 until the end of year 2000. The value 4.05 under the year 2000 means that, at the end of the year 2000, variable p1 is 4.05 percent larger than it was at the start of year 1998.

Note that, if you ask for the extra three columns (as in the section above), the values shown in these 3 columns are the same whether or not option ACC is selected.

The Windows programs RunDynam and RunMONASH (see section 2.7) produce separate spreadsheets showing year-on-year or accumulated results – you select the style you want on the Results page.

10.2.3 Accumulated Indexes (Option ACI)

If you select option ACI, the results will be accumulated as indexes relative to base 1. For more details, see the on-line help by entering option **?ACI** when ACCUM starts to run.

10.3 Using DEVIA to Prepare a Spreadsheet Table for the Differences

Whereas ACCUM can be used with any model, DEVIA is really only useful with dynamic models such as MONASH.

By default, DEVIA produces a spreadsheet file which shows the cumulative 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 year-on-year 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

$$z(j) = 100 \left[\text{Product}\{k=1,\ldots,j,\ (1 + y(k)/100)/(1 + x(k)/100)\} - 1\right]$$

for a percentage-change variable.

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

$$z(j) = \text{Sum}\{k = 1,\ldots,j,\ (y(k) - x(k))\}$$

The values z(j) calculated by DEVIA are therefore **cumulative differences to the end of the j th year**, caused by the policy shocks. [See section 10.3.2 below if you don't want the differences accumulated.]

The input to DEVIA are the file names of the year-on-year base case CSV file and the year-on-year policy deviation CSV file as produced by ACCUM. 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.]

Cumulative Differences between Policy 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. It indicates that cr is 2.08 percent larger in the deviation run at the end of year 1999 than it was in the base case at the end of the year 1999.

10.3.1 Making Sure DEVIA Knows Change/Percent-Change Variables (Option SOL)

As the formulas in section 10.3 above indicate, DEVIA needs to know whether each variable is a change variable or a percentage-change variable. You can select option **SOL** at the start of the run to tell DEVIA the name of one of the Solution files from which the results were produced. Then DEVIA can read from this Solution file which variables are changes and which represent percentage changes.

10.3.2 Year-On-Year Differences (Option NAC)

By default, DEVIA produces **cumulative** differences (as the formulas in section 10.3 above show). If you want DEVIA to show the **year-on-year differences**, select option **NAC** at the start of the run. If option NAC is selected, the values shown by DEVIA are calculated as follows.

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 (using option NAC) is

```
z(j)  = 100 [(1 + y(j)/100)/(1 + x(j)/100) - 1]
```

for a percentage-change variable. For an ordinary change variable, the value z(j) shown in the spreadsheet produced by DEVIA (using option NAC) is

```
z(j) =  y(j) - x(j)
```

The Windows programs RunDynam and RunMONASH (see section 2.7) produce separate spreadsheets showing year-on-year or cumulative differences – you select the style you want on the Results page.

Although DEVIA can produce year-on-year differences, our experience indicates that the cumulative differences are the ones you should be looking at (since they best capture the effects of the policy change). [The year-on-year differences can be confusing.] So, if in doubt, concentrate on the cumulative differences and ignore the year-on-year differences.

CHAPTER 11

11. Transferring Models Between Machines

Once you have implemented a model using GEMPACK, you or a colleague may wish to transfer the model to a different machine also running GEMPACK. The essential ingredients of a model are

- the TABLO Input file, and
- the data file(s).

Also helpful in using the model on a different machine are

- any relevant Stored-input or GEMPACK Command files, such as those for running TABLO (especially if condensation is required), for specifying the closure, or for carrying out simulations. The idea is to transfer these files and then to re-implement the model by running GEMPACK on the new machine.

Alternatively, you may wish to transfer just the ability to carry out Johansen simulations with the model. In this case you need to transfer

- the Equations file and any relevant Stored-input or GEMPACK Command files.

If you are transferring to a machine with the same operating system (for example, from one PC to another), you can probably just copy the relevant files to this machine.[38]

If, however, your target machine has a different operating system (for example, you are transferring from a PC to a unix computer or to a mainframe), copying files will not succeed. GEMPACK contains several programs (namely MKHAR, RWHAR, MKEQ, RWEQ and RWSOL, MKSOL) which make it easy to transfer models by following the procedures described below. The rest of this section is devoted to transferring between different operating systems.

When transferring between different operating systems, there are two basic principles.

1. Text files can be transferred directly.
2. Binary files must be converted to text files first.

TABLO Input, Stored-input and GEMPACK Command files are all text files. But data files for models can be either binary (if they are Header Array files) or text. Equations and Solution files are binary files.

Note that text files can be transferred between different machines in several ways including

- between any two computers by electronic transfer utilities such as FTP or Kermit,
- between mainframes using tapes (or using FTP or some similar program if a PC is an intermediary).

[38] However, this may not always be appropriate. The strict test is whether or not the machines are "binary compatible". For example, it is possible to have two different Unix machines which are not binary compatible. In such a case, follow the procedure in the rest of this section, first converting binary files to text files.

11.1 Transferring the Whole Model

Suppose that you wish to transfer a model from Machine 1 to Machine 2.

Step 1.

On Machine 1, convert any Header Array data files for the model to text files by running the GEMPACK program RWHAR. (Run it once for each such Header Array file. The program requires just two inputs, firstly the name of the existing binary Header Array file and secondly the name of the text file to be created.)

Step 2.

Transfer the TABLO Input file, the text versions of any data files and any Stored-input or GEMPACK Command files to Machine 2.

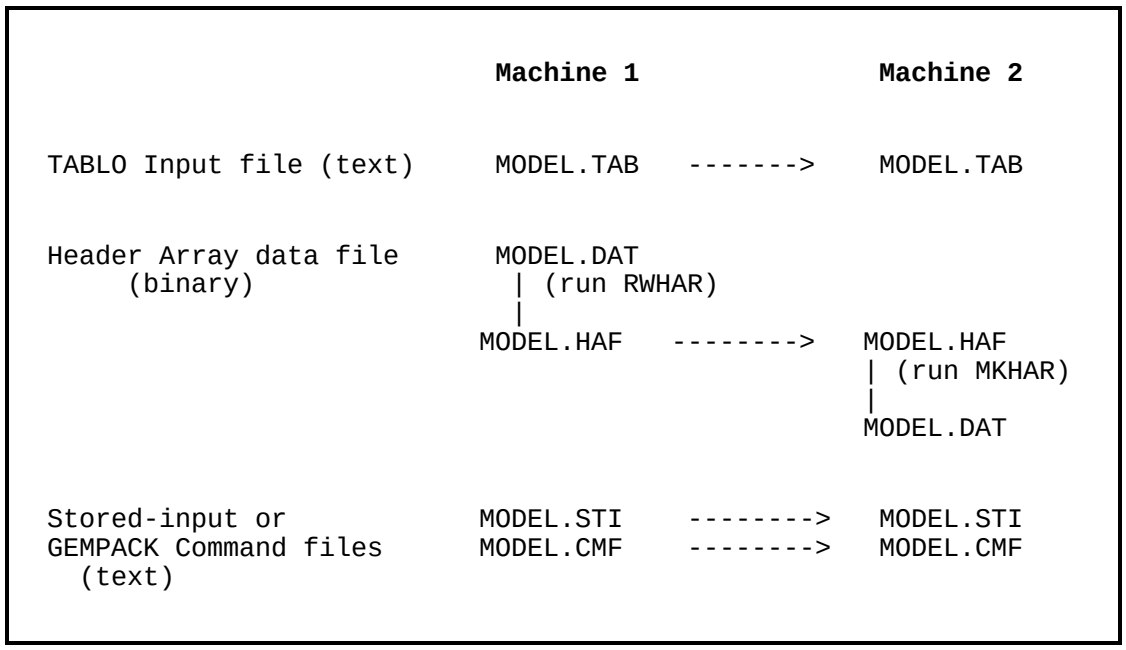
Step 3.

On Machine 2, convert any text versions of Header Array files back to binary files by running the GEMPACK program MKHAR. (Run it once for each Header Array file. The program requires just two inputs, firstly the name of the text version of the Header Array file and secondly the name of the binary Header Array file to be created.)

Step 4.

Re-implement the model on Machine 2 by running TABLO etc.

Below we illustrate this for a model with just one data file MODEL.DAT which is a Header Array file. In this diagram "--->" indicates file transfer between the machines.



11.2 RWSOL, MKSOL: Transferring Solution Files

Unix users have asked us about using the Windows ViewSOL (see section 2.3) for looking at simulation results in Solution files. We have provided two programs **RWSOL** and **MKSOL** for transferring Solution files between machines. For example, to transfer a Solution file from a Unix machine to a Windows PC, run RWSOL on the unix machine to convert the Solution file to a text file. Then transfer this text file (for example, using FTP) to the PC. Run MKSOL on the PC to convert this to a binary Solution file on the PC; you can use ViewSOL to look at this.

If you want to be able to run AnalyseGE on a PC to analyse simulation results produced on a computer other than a PC (for example, a Unix machine), you must transfer the SLC file[39] as well as the Solution file. Since the SLC file is a Header Array file, you can use RWHAR and MKHAR (see section 11.3).

11.3 RWHAR, MKHAR: Transferring Data Files

In a similar way if you are working on a Unix machine but wish to look at your Header Array files using the Windows program ViewHAR, use RWHAR to convert the Header Array file on your Unix machine to a text file, transfer the text file to the PC (by FTP). Then on the PC, run the program MKHAR to convert it back to a Header Array file. Then you will be able to use ViewHAR on the PC version.

11.3.1 RWHAR - Options CPR, C51

RWHAR has an 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 option

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

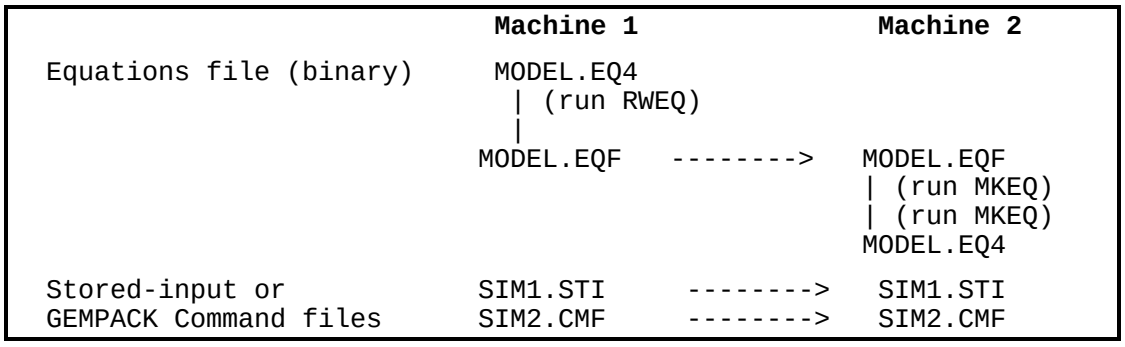
allows you to remove any set and element labelling from a Header Array file. In this case the output from RWHAR is a Header Array file (not a text file, as usual). See section 5.4 for more details about this option.

[39] SLC files are described in section 8.4 of GPD-3. SLC files are new for Release 7.0 of GEMPACK.

11.4 Transferring Johansen Simulation Capability

This is rarely done now since more powerful computers make the running of simulations relatively painless. So it is usual to transfer the whole model rather than the Equations file.

Because each Equations file is a binary file, it must be converted to a text file on Machine 1, then transferred, and finally converted back to a binary file on Machine 2. The GEMPACK program RWEQ converts a binary Equations file to a text file while the program MKEQ converts the text version of an Equations file back to a binary file.



11.5 Comparing Results from Different Machines

The programs COMPEQ, RWSL and CMPSOL were developed to check that, when GEMPACK is moved to machines other than those on which it was developed, it is producing correct numerical results. The different arithmetic on different machines is almost certain to produce slightly different results but we need to check that any differences found are within the limits expected.

COMPEQ is for comparing two Equations files (usually one produced on the machine in question and the other moved from our development machine after first converting it to a text file using the program RWEQ). It tells how different the numerical values are.

RWSL is for converting simulation results (on a Solution file) to a text file. Then they can be moved to another machine and compared (using CMPSOL) with the results obtained on the new machine.

As with CMPHAR (see section 4.2 above) these programs report totals, averages and largest absolute differences and difference ratios.

It is unlikely that modellers will need to use these programs. Any user requiring more information about them should consult us at the Impact Project.

CHAPTER 12

12. SEENV: To See the Closure on an Environment File

SEENV is a 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 three possible types are:

- text suitable for including in a Command file to give the closure,
- a text file which can be used as a Spreadsheet Mapping file (see section 9.1),
- a text file containing Shock statements for the bottom of a Command file such as **Shock x1 = select from file xx.har header "ABCD" :**

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).

12.1 Command File Output

If you ask for Command file output and choose **exogenous** variables, the output will 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>
  ;
```

12.2 Spreadsheet Output

For spreadsheet output, the syntax is that used in Spreadsheet Mapping files (see section 9.1). 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 Mapping file 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 Mapping file contains just the endogenous variables and the endogenous components of variables.

If you choose **both**, the Spreadsheet Mapping file 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. If one variable has some components exogenous and other components endogenous, the variable will feature twice in the output, its exogenous components being listed amongst the exogenous variables and its endogenous components being listed amongst the endogenous variables.

12.3 Shock Statements for the Bottom of a CMF File[40]

If you have a file containing the Environment (for example an Environment or Solution file) and a SOL file (a Header Array file produced from a Solution file by SLTOHT), program SEENV writes a shock statements for each exogenous variable on the Environment file.

`Shock` <variable_name> = `select from file` <SOL_filename> `Header` "<header_name>" ;

Here <variable_name> is the variable taken from the Environment file, and <header_name> is the appropriate header on the SOL file where the values of the shock are stored. "Select from" is used in case not all the components of the variable are exogenous.

This technique can be used in the RunDynam software (see section 2.7) when rerunning the Base case with the policy closure. Here the SOL file is the Header Array file made from the Solution file from the Base Case solution. The Environment file is the Environment for the Policy (or Deviation) closure. If you want to do this under RunDynam, select the menu item ***Options | Use SOL files for shocks***.

In cases where you need to read a large number of shocks to rerun the Base Case, this may be faster than reading all the shocks one by one from simple shock statement in the Command file.

[40] This possibility was introduced in Release 7.0.

CHAPTER 13

13. SUMEQ: Information from Equations Files

The Equations file for a model is essentially just a computer encoding of the Equations Matrix C in the system of linearized equations

$$C\ z = 0 \qquad (1)$$

for the (condensed) model, as described in section 2.11.1 in GPD-1. Here the Equations Matrix C is of size n x m where n is the total number of equations and m is the total number of variables in the (condensed) model.

As we have seen in section 2.11.1 in GPD-1, it is often useful to think of the matrix C as a rectangular array or tableau with the vector variables (that is, the VARIABLEs in the TABLO Input file) across the top and the equation blocks (that is, the EQUATIONs from the TABLO Input file) along the left-hand side. Each vector variable occupies as many columns as its number of components, and each equation block occupies as many rows as the number of actual equations in it. Together the groups of columns and rows divide the tableau into a block pattern.

Part of the tableau for the 27 x 29 Equations Matrix C for Stylized Johansen is shown in Table 2.11.1 in GPD-1.

Notice that this divides up the Equations Matrix C into rectangular **submatrices**, where the columns and rows of each submatrix correspond respectively to components of the vector variable and the equations in the equation block determining this submatrix. Only some of these submatrices can contain nonzero entries. For example, the equation block "House" only involves the vector variables p_XH, p_Y and p_PC, so the submatrices in these rows corresponding to the other vector variables must be filled with zeros. Indeed, this is why, in general, the Equations Matrix of a general equilibrium model is sparse.

You have probably noticed that, when GEMSIM or a TABLO-generated program for a model is calculating the equations (that is, working out the numbers in the Equations Matrix for the model), it tells you how many nonzero entries are in each of the submatrices for each equation in turn.

The program SUMEQ (**SUM**marise an **EQ**uations file) can be used to obtain various pieces of information about an Equations file.

- It can produce a **map** of the Equations Matrix C, showing the variables and equations and the columns and rows in the Equations Matrix to which they correspond. This is described in more detail in section 13.1.1 below.
- It can produce information about row and column sums (across selected columns or rows, if required). This can be especially useful since it turns out to be a very good way of checking the implementation of many models.
- You can use it to examine individual entries in the Equations Matrix. You specify the row and column and SUMEQ tells you the value of the entry in that row and column of the Equations Matrix. The map produced by SUMEQ (see section 13.1.1 below) will be helpful in deciding which rows and columns you are interested in.

SUMEQ is easy to run interactively. Its output is always directed to a Print file.

13.1.1 Map of an Equations File Produced via SUMEQ

SUMEQ is often used to produce a map of the Equations file. If you run SUMEQ in WinGEM, it produces a map which shows

- the sets and their sizes,
- the vector variables, the number of components in each and the corresponding columns in the Equations Matrix,
- the equation blocks, the number of equations in each and the corresponding rows in the Equations Matrix.

Below we show this map for the Stylized Johansen model (the version whose TABLO Input file is shown in section 3.3.2 in GPD-1). Note that, in this map, the "descriptions" of the sets, variables and equation blocks are taken directly from the labelling information (between hashes '#') on the TABLO Input file; sometimes this is truncated in the map.

MAP OF VARIABLES AND EQUATIONS FOR STYLIZED JOHANSEN PRODUCED BY SUMEQ

```
  SETS
  ----

  No Name          Size    Description
-------------------------------------
    1 SECT             2   Sectors
    2 FAC              2   Factors
    3 NUM_SECT         1   Numeraire sector - sector 1

  VARIABLES
  ---------
  No Name     Size Cols      Arguments (if any) and Description
-------------------------------------------------------------------------
   1 p_Y         1 1           Total nominal household expenditure
   2 p_PC        2 2-3       (SECT)  Price of commodity i
   3 p_PF        2 4-5       (FAC)  Price of factor f
   4 p_XCOM      2 6-7       (SECT)  Total demand for commodity i
   5 p_XFAC      2 8-9       (FAC)  Total demand for factor f
   6 p_XH        2 10-11     (SECT)  Household demand for commodity i
   7 p_XC        4 12-15     (SECT,SECT)  Intermediate inputs of c ...
   8 p_XF        4 16-19     (FAC,SECT)  Factor inputs to industry j
   9 p_DVCOMIN   4 20-23     (SECT,SECT)  Dollar value of inputs o ...
  10 p_DVFACIN   4 24-27     (FAC,SECT)  Dollar value of factor f  ...
  11 p_DVHOUS    2 28-29     (SECT)  Dollar value of household use ...

  EQUATIONS
  ---------
  No Name          Size Rows      Arguments (if any) and Description
-----------------------------------------------------------------------
  1 Comin             4 1-4       (SECT,SECT)  Intermediate input  ...
  2 Facin             4 5-8       (FAC,SECT)  Factor input f to in ...
  3 House             2 9-10      (SECT)  Household demand for com ...
  4 Com_clear         2 11-12     (SECT)  Commodity market clearing
  5 Factor_use        2 13-14     (FAC)  Aggregate primary factor  ...
  6 Consumer_demands  2 15-16     (SECT)  Household expenditure fu ...
  7 Intermediate_com  4 17-20     (SECT,SECT)  Intermediate demand ...
  8 Factor_inputs     4 21-24     (FAC,SECT)  Factor input demand  ...
  9 Price_formation   2 25-26     (SECT)  Unit cost index for indu ...
 10 Numeraire         1 27        (NUM_SECT)  Price of commodity 1 ...
```

You might like to look back at the tableau shown in section 2.11.1 of GPD-1 above to see the significance of the entries listed under the headings "Cols" and "Rows" in the map just above. You might like to run SUMEQ on the Equations file (probably called SJ.EQ4) you produced in Step 3 in section 2.10.1 of GPD-1. If you select the map option 'm', you should obtain the map shown above.

Note that, if you have condensed your model, only VARIABLEs and EQUATIONs left in the condensed system appear on the Equations file and hence in the map produced by SUMEQ.

In order to identify the column corresponding to a particular component of a variable, you will also need to know the order of the components, which is documented in section 5.3 of GPD-3. To identify the row corresponding to a particular equation, you will also need to know the order of the equations in a block, which is documented in section 4.14.5 of GPD-2.

13.2 SUMEQ and the Homogeneity of Models

This section introduces the topic of homogeneity simulations.

Many general equilibrium models have various so-called homogeneity properties, for example that one solution of the model is obtained by increasing all nominals (such as domestic prices, domestic dollar values and the exchange rate) by 1 per cent, while holding all reals (such as physical quantities) unchanged. Checking such homogeneity properties is one important way of verifying that you have implemented your model correctly.

The program SUMEQ can be used to check the homogeneity of models. We include a detailed example in section 13.2.2 below.

Suppose you have a linearised equation

```
A*p(1) + B*x(1) + C*p(2) + D*x(2) + ... = 0.
```

If you 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.

When reporting row sums, SUMEQ suppresses[41] 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 easy to identify equations which are not homogeneous. [To find which All arguments correspond to a particular equation number within a block of equations, you will also need to know the order of the equations in a block. This is documented in section 4.14.5 of GPD-2.]

In general, to see if your model is homogeneous, use SUMEQ to calculate all row sums of the Equations Matrix across just those columns which correspond to the variables which should increase by 1 per cent. (Use the map of the Equations file to identify which column numbers these are.) All row sums should equal zero (to within rounding errors). To identify a collection of columns, you can input

7-10, 30, 35-38

as an abbreviation for 7,8,9,10,30,35,36,37,38.

13.2.1 Why Using SUMEQ is Better than Carrying out a Simulation

Note that you could also check such homogeneity properties by actually carrying out the simulation in question (via SAGEM, for example). If you have made an error in implementing your model (for example, a typographical mistake in your TABLO Input file), the simulation results will tell you something is wrong, but you will not know where to look to correct the error. The advantage of using SUMEQ is that the printout of the row sums produced will identify the problem by showing you exactly which equation block(s) in your TABLO Input file are incorrectly specified (since only these

[41] This was new for SUMEQ (version 1.7, April 1996) distributed with Release 5.2

row sums will not be as expected). (Use the map produced by SUMEQ to find which equation block each actual equation is in.) However, if you have condensed your model, it may be more difficult to spot which equation block is in error, since if an equation block with an error in it is substituted out, it may contaminate all equation blocks into which it is substituted.

13.2.2 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 13.1.1). For Miniature ORANI as in MO.TAB distributed with GEMPACK, part of such a map is shown below. [The complete map contains as well a list of Equations and the Rows.]

```
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.

Run SUMEQ interactively. 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.]

13.2.3 SUMEQ for Homogeneity - Summary

The example above (section 13.2.2) 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.

CHAPTER 14

14. TEXTBI : Extracting Text Files from Binary Files

The TABLO Input file of the model is stored

- on the Auxiliary Table file (produced when TABLO runs)
- on the Solution file (produced when GEMSIM or the TABLO-generated program runs).

[See sections 2.1.1 and 2.1.2 of GPD-2 and section 8.2.1 of GPD-3 for more details.]

The Command file from the simulation is stored on the Solution file (see section 8.2.2 of GPD-3).

The Stored-input file used to condense the model (if condensation is carried out when running TABLO) is stored

- on the Auxiliary Table file (produced when TABLO runs)
- on the Solution file (produced when GEMSIM or the TABLO-generated program runs).

[See sections 2.1.1 and 2.1.2 of GPD-2 and section 8.2.1 of GPD-3 for more details.]

The program TEXTBI can be used to recover any one of these files from the binary file on which it is stored. For example, to recover the TABLO Input file from a Solution file, run TEXTBI, select the default program options, then respond ‘t’ to the prompt asking which file you wish to recover, and then specify the name of the Solution file and the name of the recovered TABLO Input file.

CHAPTER 15

15. Converting Binary Files: LF90/F77L3 and LF95

On Windows PCs, three Fortran compilers are supported, namely F77L3, LF90 and LF95 (all produced by Lahey Computer Systems, Inc.). LF95 was not supported before Release 7.0 of GEMPACK.

Binary files (most importantly, Header Array files) produced by F77L3 can be read by LF90 programs and vice versa. Unfortunately this compatibility does not extend to LF95. Binary files produced by LF95 programs cannot be read directly by programs compiled and linked using LF90 or F77L3. Similarly, binary files produced by LF90 or F77L3 programs cannot be read directly by programs compiled and linked using LF95.

However we supply with GEMPACK tools which allow you to convert binary files of one of these types to the other type (see section 15.1 below). Indeed it is not absolutely necessary to convert the files since Release 7.0 (or later) GEMPACK programs produced using any one of LF90, F7L3 or LF95 are able to read any of these files because they first convert the file at run time if needed (see section 15.2 below).

First we need to settle on some terminology for discussing this issue.

We call binary files produced by programs compiled and linked using LF90 or F77L3 **Lahey files**.

We call binary files produced by programs compiled and linked using LF95 **Fujitsu files** (since LF95 is derived from a suite of Fortran compilers developed by the Fujitsu company).

15.1 Converting Between Lahey and Fujitsu Binary Files

The GEMPACK program **CONLF** can be used to convert a Lahey file to a Fujitsu one or vice versa. To convert a Lahey file to a Fujitsu one, select the program action **CL** (Convert Lahey). To convert a Fujitsu file to a Lahey one, select the program action **CF** (Convert Fujitsu).

The Windows program **ConvHAR** (written by Mark Horridge) can be used to indicate which files in a directory are Lahey files and which are Fujitsu files. It can also be used to convert all Lahey files to Fujitsu files, or vice versa. This program ConvHAR is available from the GEMPACK web site if you need to use it.

Recent versions of ViewHAR can read both Lahey and Fujitsu files, and can save in either Lahey or Fujitsu format. See the ViewHAR Help file for more details.

15.2 GEMPACK Programs on PCs Can Handle Both Type of Files

Release 7.0 (or later) GEMPACK programs compiled and linked with any of LF90 , F77L3 or LF95 can usually handle either Lahey or Fujitsu files. They do this by making a converted version of any files of the wrong type at run time.

For example, suppose you have a Source-code version of GEMPACK and are using LF95. Suppose also that you are running the TABLO-generated program SJ.EXE you have produced (using LF95) and ask it to read a Lahey version of the base data file SJ.DAT. Your program SJ.EXE will recognise that this data file SJ.DAT is of the wrong type to be read directly and will convert the file to a Fujitsu file containing the same data. It will then read the Fujitsu version (and will ignore the Lahey version). Of course, the updated version of this data produced by the simulation will be a Fujitsu file (since it is an LF95 program running).

This ability of Release 7.0 (or later) GEMPACK programs means that you do not always have to convert binary files before using them. However if you have Lahey programs (produced via LF90 or F77L3) and you receive lots of Fujitsu files from a colleague, it is probably best to convert them (following one of the methods described in section 15.1 above) before using them. Otherwise, your programs will continually be taking small amounts of time making copies of the data and will be leaving these temporary copies around to fill up your hard disk.

Note however that Release 6.0 or earlier GEMPACK programs (these must have been compiled and linked using LF90 or F77L3) cannot read Fujitsu binary files.

15.2.1 Names and Location of These Converted Files

When a GEMPACK program converts one of these files to another of the opposite kind (for example, converts Lahey SJ.DAT to a Fujitsu version of the same data, as in the example above), you may need to know

- how the file name of the copy is made up and,
- in which directory the copy is placed.

The file name is of the form **Gdddeeee.TMG** where

1. ddd is the number of the current day in the year (for Example, January 1 corresponds to "001" while December 31 corresponds to "365" except in a leap year when it corresponds to "366"),
2. eeee is the number of this copy made on this day.

So, for example, the first converted file made on January 1 will be called G0010001.TMG while the twentieth converted file made on the same date will be called G0010020.TMG.

F77L3 always puts these temporary copies in the current directory (wherever the program is running). LF90 and LF95 put these files into the directory pointed to by the Environment variable **TEMP** on your computer. [Often this points to directory C:\WINDOWS\TEMP under Windows 95 or 98.] If TEMP is not set, LF90 and LF95 put these files into the current directory (wherever the program is running).

Clean Up These Files Periodically

Most users will rarely (if at all) make use of the program's ability to convert files at run time, so can ignore this advice. However, if you know that file conversions are being made regularly on your PC, you should take the trouble to periodically delete old temporary files. The information above about their names and location should make that easy.

CHAPTER 16

16. Translation from GEMPACK to GAMS Data Files

GAMS [Brooke *et al* (1988) and Rutherford (1999)] is used by many modellers for general equilibrium modelling.

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 7.0 (or later) of GEMPACK, there are also other options (including **GM1**, **GM2** and **GM3**) 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. Choosing Sets of Variables Interactively[42]

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 5.4.3 of GPD-3 and in section 7.3 above. 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 are described in section 17.1 below. 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 is described in detail in section 17.3 below while the "responding to prompts" method is described in section 17.4 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.

---

[42] 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 4.4 of GPD-1, the menu will be compressed considerably.]

Choices a,n,m,1,f,w are described in section 17.1 below, choice L (Lists) is described in section 17.3 and choice s (SOME) is in section 17.4 below.

17.1 Simple Choices

The simple choices, and their meanings are shown below.

Select

```
a (ALL)     if the subset is identical to the big set.
n (NO)      if the subset contains no variables.
m (MACROS)  if the subset consists of all macro variables
            which are in the big set. A macro variables has
            exactly one component.
```

If you select one of these menu items, this completes the choice of subset. (If you want to choose all macro variables **and** some other variables, choose 'L' as in section 17.3.)

Select

```
1 (ONE)     to choose all components (which are in the big set)
           of just one variable. You will then be asked for
            the name of this one variable.  .
```

Note that only the components of a variable which are in the big set are available for the subset. If, for example, a variable 'x1' has 10 components of which only numbers 1-5 and 10 are in the big set and you select '1' and then nominate 'x1', only these 6 components will be in the subset.

Select

```
f (FEW)   to choose all components (which are in the big set)
          of a few variables. You will then be asked how many
          variables and then for the names of these variables
          (one per line).
w (WHICH) for a description of which components of which
          variables are in the big set. Then you can choose
          again from the menu above.
```

17.2 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)

[See case 1) in section 5.4.1 of GPD-3 for more details about this.]

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 5.3 of GPD-3 for the order of the components of variables with two or more arguments.

17.3 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?
f	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.

First we give two examples to help to make the procedure clear.

EXAMPLE 1

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 17.2 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.

EXAMPLE 2 (Miniature ORANI)

In the Miniature ORANI model (see section 5.1.1 of GPD-3 and section 13.2.2 above), suppose we wish to choose

1. all components of p_Z,
2. just components p_YCOMIND(c,"i2") for all 'c' in COM,
3. components 1,2,3 of variable p_XINTFAC, components 1 and 4 of p_XINTCOM,
4. all macro variables, and
5. all variables with one argument where this argument belongs to the set COM. Also all variables with two arguments, the first belonging to set IND and the second to FAC.

The responses for these choices are as follows:

1. To choose all components of p_Z (that are in the big set), enter

   a p_Z

2. The response below will choose the desired components of p_YCOMIND.

   a p_YCOMIND(COM,"i2")

3. After this, a response of

   s p_XINTFAC p_XINTCOM

   followed by, (responding to prompts about p_XINTFAC and p_XINTCOM)

   3 (for 'p_XINTFAC')
   1-3
   2 (for 'p_XINTCOM')
   1 4

   will say that just components numbered 1-3 of 'p_XINTFAC' and components 1 and 4 of 'p_XINTCOM' are to be chosen.

4. To choose macro variables, enter
   m

5. You can choose whole classes of variables by indicating their arguments and the sets over which they range. For example

   a (COM) (IND,FAC)

   chooses all (available) components of
   (i) all variables in the model having one argument ranging over the set COM (this is what **(COM)** means), and
   (ii) all variables in the model having two arguments ranging respectively over the sets IND and FAC (this is what **(IND,FAC)** 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.

6. Then give the command which completes the choice of the subset.
   f (to finish)

17.3.1 Lists Option - The Subcommands in More Detail

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.

Below we go through all the subcommands and give details about each.

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 17.2 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.

17.4 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 17.2 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.

18. REFERENCES

Brooke, T., D. Kendrick and A. Meeraus (1988), *GAMS: A User's Guide*, The Scientific Press, Redwood City, California.

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., and M.T.Rimmer (2000), *Forecasting, Policy, History and Decomposition: the MONASH Model of the Australian Economy*, (Draft, October 2000), Centre of Policy Studies and Impact Project, Monash University.

Pearson, K.R., Thomas W. Hertel and J. Mark Horridge (2000), *AnalyseGE: Software Assisting Modellers in the Analysis of their Results*, June 2000. [The current version of this document can be downloaded from the AnalyseGE page on the GEMPACK web site.]

Rutherford, Thomas .F. (1999), 'Applied General Equilibrium Modeling with MPSGE as a GAMS Subsystem: An Overview of the Modeling Framework and Syntax', *Computational Economics*, vol. 14, pp. 1-46.

19. GEMPACK DOCUMENTS[43]

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

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

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

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

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

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

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

[43] 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".

20. INDEX

1

1C. *See* Array types

2

2I. *See* Array types
2R. *See* Array types

A

Absolute difference, 4-35
ACCUM, 10-83
Accumulated differences, 10-86, 10-87
Accumulated indexes, 10-86
Accumulated results, 10-85
Aggregating data
  Using ViewHAR, 2-4
AnalyseGE, 2-9
  Transferring files, 11-91
Array sizes
  Text file, 6-47
Array types, 3-11, 5-38
  1C, 3-11
  2I, 3-11
  2R, 3-11
  RE, 3-11
  RL, 3-11

B

Binary files
  Fujitsu, 15-103
  Lahey, 15-103
  Transfer of files, 11-89

C

Center for Global Trade Analysis, 2-8
Character strings
  Text file, 6-49
Checking models, 13-97
Choice of variables
  Interactive, 7-56
  Macros,all,1, 17-110
  Sets, 17-112, 17-113
Closure
  Seeing, 12-93
CMBHAR, 4-36
CMPHAR, 4-35
Coefficient
  How much data information, 5-40
Column order
  SLTOHT, 8-68
  Text file, 6-48
Column sums
  Equations, 13-95
Comments
  Text file, 6-45
Comparing
  Header Arrays, 4-35
  Solution files, 4-35
Compatibility
  Release 5.1, 5-42
Components of variables
  Specifying, 17-111
Compressed form
  RWHAR, 11-91
CONLF, 15-103
  Converting between Lahey and Fujitsu files, 15-103
ConvHAR, 15-103
  Converting between Lahey and Fujitsu files, 15-103
CSV file
  GEMPACK text file, 6-51
CSV format, 4-32
Cumulative solution, 8-66

D

Data
  Comparing, 4-31
Data manipulation, 8-59
Database
  SQL, 4-34
Define sets
  MODHAR, 3-20, 5-41
DEVIA, 10-83
  Cumulative differences best to use, 10-87
Difference ratio, 4-35
Different machines
  Binary compatible, 11-89
Displays, 4-33
Dynamic models, 10-83

E

Element names
  SLTOHT, 9-76
  Text file, 6-51
Environment
  SEENV, 12-93
Equations file
  Contents, 13-95
Equations matrix
  Map, 13-95
Errors rounding, 4-35

F

File
  GAMS, 16-105
Forecasting models, 10-83
FTP, 11-89
Fujitsu files, 15-103
  Converting to Lahey, 15-103
  Definition, 15-103

G

GAMS, 16-105
  Translation, 16-105
GAMS output, 4-33
GEMPACK programs, 1-1
GEMPIE
  Levels results, 7-54
  Subtotals results, 7-54
  Zero results, 7-56
GEMSIM
  Set and element information, 5-39
Getting started with GEMPACK, 2-10
Graphs
  Using SLTOHT, 9-82
GTAP, 2-8
GTEM model, 10-83

H

Header Array
  Transfer, 11-89
Header Array file, 4-31
  From Solution, 8-59
Header Array files
  Fujitsu, 15-103
  Lahey, 15-103
Header Mapping file
  SLTOHT, 8-66
Homogeneity, 13-97
  Example, 13-97
  SUMEQ, 13-97
How much data information
  Coefficient, 5-40

I

Individual Column results
  Printing, 7-55
Individual solutions, 8-66

K

Kermit, 11-89

L

Labelled output
  From SEEHAR, 4-31
Labelling
  Set and element, 5-37
Lahey files, 15-103
  Converting to Fujitsu, 15-103
  Definition, 15-103
Levels results, 8-66
Levels solutions
  SLTOHT, 8-65
Levels values
  SLTOHT, 9-78
Licence file, 1-2
Line length
  Text file, 6-50

M

Macro variables, 7-56, 17-109, 17-113
Map of Equations file, 13-96
Mapping file
  Header, 8-66
  SLTOHT Header Array, 8-66
  Spreadsheet, 9-73
Mapping files
  SLTOHT, 8-63
MKEQ, 11-89
MKHAR, 11-89
  Set end element information, 5-39
MKSOL, 11-89
Model verification, 13-97
MODHAR
  Defining sets, 3-20, 5-41
  Set and element information, 5-38, 5-40
MONASH model, 8-70, 10-83
  SLTOHT option SHK, 8-70
MS Access, 4-34

O

Options
  ACCUM - ACC, 10-85
  ACCUM - ACI, 10-86
  CMPHAR SOL, 4-35
  DEVIA - NAC, 10-87
  DEVIA - SOL, 10-86
  GEMPIE - CPL, 7-57
  GEMPIE - CPW, 7-57
  GEMPIE - FBS, 7-57
  GEMPIE - NBV, 7-57
  GEMPIE - OSR, 7-57
  GEMPIE - OSS, 7-57
  GEMPIE - RPO, 7-57
  GEMPIE - SNA, 7-57
  RWHAR - C51, 5-42, 11-91
  RWHAR - CPR, 11-91
  SEEHAR - COL, 4-32
  SEEHAR - CPL, 4-33
  SEEHAR - CPW, 4-33
  SEEHAR - DEC, 4-33
  SEEHAR - FNP, 4-33
  SEEHAR - GMS (etc), 4-33
  SEEHAR - NEL, 4-32
  SEEHAR - ROW, 4-32
  SEEHAR - SQL, 4-34
  SEEHAR - SS, 4-32
  SEEHAR - SSE, 4-32, 5-37
  SLTOHT - LV, 8-70
  SLTOHT - NEL, 8-70, 9-76
  SLTOHT - NLV, 8-65
  SLTOHT - SEP, 8-70
  SLTOHT - SHK, 8-70
  SLTOHT - SHL, 8-65
  SLTOHT - SIC, 8-68
  SLTOHT - SIR, 8-68
  SLTOHT - SS, 8-68
  SLTOHT - SSE, 9-77
  SLTOHT - SSS, 9-76
  SLTOHT VAI, 8-67
Oracle, 4-34

P

Page width,length
  GEMPIE, 7-57
Prompts
  Some option, 17-114

R

RE. *See* Array types
RE type (data on HA file), 5-38
Real arrays
  Multi-dimensional, 6-49
Real Numbers
  Text file, 6-48
Recursive dynamic models, 8-70
  SLTOHT option SHK, 8-70
Repeated values
  Text file, 6-49
RL. *See* Array types
Row order
  SLTOHT, 8-68
  Text file, 6-48
Row print order
  GEMPIE, 7-57
Row sums
  Equations, 13-95
  Homogeneity, 13-97
RunDynam, 2-10, 10-84, 12-94
RunGEM, 2-7
RunGTAP, 2-8
RunMONASH, 2-10, 10-84
RWEQ, 11-89
RWHAR, 11-89
  Set end element information, 5-39
RWSOL, 11-89

S

SEEHAR, 4-31, 4-33
  Labelled output, 4-31
  Set and element labels, 5-37, 5-38
SEENV, 12-93
Sensitivity analysis. *See* Systematic sensitivity analysis
Set and element information, 5-37
  Adding, 5-39
  Adding in MODHAR, 5-40
  Checking when reading data, 5-39
  Removing, 5-42, 11-91
Set and element labelling
  Types RE/RL/2R, 5-38
Sets of variables
  Choosing, 17-112, 17-113
Shocks
  SLTOHT, 8-65
Shocks for Command files
  SLTOHT, 8-70
Short spreadsheet output
  SLTOHT, 9-76
Significant difference, 4-35
Simulation results
  Analysing, 2-9
SLC file
  Transferring, 11-91
SLTOHT, 8-59
  Levels solutions, 8-65
  Spreadsheet Mapping file, 9-73
  Spreadsheet Mapping files, 9-73
SLTOHT Mapping files
  Rules, 9-80
Solution
  To Header Array, 8-59
  To Text file, 8-59
Solution files
  Transferring between machines, 11-91
Sparse matrix, 13-95
Spreadsheet, 4-32
  Example, 8-69
  From Solution, 8-59
  Preparing tables, 9-73
  SLTOHT, 8-68
  Text file, 6-48
Spreadsheet Mapping file, 9-73, 12-94
  Example, 8-63
  Examples, 9-73
Spreadsheet output, 4-33
Spreadsheet separator, 4-33
SQL, 4-34
SS2SHOCK, 8-70
SSA. *See* Systematic sensitivity analysis
Submatrix, 13-95
Subtotal
  Printing, 7-55
Subtotal solution, 8-66
SUMEQ, 13-95
  Example, 13-97
  Homogeneity, 13-97
SUMHAR, 4-36
Systematic sensitivity analysis, 2-8

T

Tables
  3-dimensional, 9-78
  Several variables, 9-79
  SLTOHT - SSE, 9-77
TABLO-generated programs
  Set and element information, 5-39
TABmate, 2-6
TABmate plus RunGEM
  Alternative to WinGEM?, 2-8
Text file
  Data values, 6-47
  How much data information, 5-40
  Line length, 6-50
  Spreadsheet CSV, 6-51
Text files
  From Solutions, 8-59
  Transfer of files, 11-89
TEXTBI, 14-101
Transferring data
  Between PCs and other computers, 11-89
Transferring models, 11-89
Transferring Solution files to PCs, 11-91

U

Undefined Solution values
  SLTOHT, 8-65
Updated data
  Set and element information, 5-39

V

Variables
  Macro. *See* Macro variables
  Specifying components, 17-110
ViewHAR, 2-4
ViewSOL, 2-5, 11-91
  Not showing all variables?, 2-5
  Results shown not up-to-date?, 2-5

W

Windows programs
  Not under Windows 3.1, 2-3
  Run under Windows 95, 98, NT or 2000, 2-3
  RunDynam and RunMONASH, 10-84
WinGEM, 2-3, 2-10

Y

Year-on-year differences, 10-87
Year-on-year results, 10-85