# GEMPACK USER DOCUMENTATION

**Release 8.0**

**GPD-1 An Introduction to GEMPACK**

# An Introduction to GEMPACK

**GEMPACK Document No. GPD-1**

**W. J. Harrison**

**K. R. Pearson**

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

**Sixth edition**
**October 2002**

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

Sixth edition
October 2002

ISSN 1030-2514

ISBN 0-7326-15046

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

# Abstract

GEMPACK is a suite of general-purpose economic modelling software designed for solving applied general equilibrium models. It can handle a wide range of economic behaviour. It also contains powerful capabilities for solving intertemporal models.

GEMPACK has two key strengths. Firstly, it enables modellers to solve very large systems of non-linear equations. The GEMPACK software calculates accurate solutions of the economic model. Secondly, once the model equations have been specified using an algebra-like notation, modellers are freed from the computing details of the solution process.

GEMPACK runs on a wide variety of computers including

- Pentium or similar PCs running Windows 95, 98, ME, NT, 2000 or XP,
- Unix machines, and
- other mainframe, mini and microcomputers with an ANSI standard Fortran 90 compiler.

The software is provides a range of utility programs for handling the economic data base and the results of simulation. Windows programs are supplied for data manipulation, solution viewing and developing your model equations. A recent development is a Windows program that aids simulation analysis and interpretation of simulation results.

GEMPACK software is used in over 300 organisations in over 60 countries around the world.

GEMPACK is fully documented with plenty of examples, including a range of example models. This document contains a guide to the rest of the GEMPACK documentation.

This document is an introduction to GEMPACK. It aims to cover the main aspects of using GEMPACK software in a practical way. In particular, it describes how to build and/or modify models and how to carry out simulations.

Document Attributes

Name : An Introduction to GEMPACK

Audience : CGE Modellers

Identifier : GPD-1

| History : | Date | Author(s) | Comment |
|---|---|---|---|
| | April 1993 | Jill Harrison and Ken Pearson | First Edition (Release 5.0) |
| | April 1994 | Jill Harrison and Ken Pearson | Second Edition (Release 5.1) |
| | Sept 1996 | Jill Harrison and Ken Pearson | Third Edition (Release 5.2) |
| | October 1998 | Jill Harrison and Ken Pearson | Fourth Edition (Release 6.0) |
| | October 2000 | Jill Harrison and Ken Pearson | Fifth Edition (Release 7.0) |
| | October 2002 | Jill Harrison and Ken Pearson | Sixth Edition (Release 8.0) |

# Table of Contents


**1. INTRODUCTION 1-1**

**1.1 Documentation of GEMPACK 1-2**

**1.2 Contacting the Centre of Policy Studies / Impact Project 1-3**

**1.3 GEMPACK World-Wide Web Site 1-3**

**1.4 GEMPACK-L Mailing List 1-3**

**1.5 A Guide to The GEMPACK Documentation 1-5**
1.5.1 For New GEMPACK Users – Getting Started 1-5
1.5.2 Guide For Experienced GEMPACK Users 1-6

**1.6 New Features in Release 8.0 of GEMPACK 1-7**
1.6.1 Main New Features 1-7
1.6.2 TABLO Input files 1-7
1.6.3 Running TABLO 1-8
1.6.4 Command files for TABLO-generated Programs and GEMSIM 1-8
1.6.5 Simulations with TABLO-generated Programs and GEMSIM 1-9
1.6.6 Windows Programs 1-10
1.6.7 Utility Programs 1-10
1.6.8 All GEMPACK Programs 1-11

**1.7 The GEMPACK Programs 1-12**
1.7.1 TABLO-Generated Programs 1-13

**1.8 Models Supplied with GEMPACK 1-14**

**1.9 Different Versions of GEMPACK and Associated Licences 1-15**
1.9.1 Source-code Versions and Licences 1-15
1.9.2 Limited Executable-image Version and Licence 1-15
1.9.3 Unlimited Executable-image Version and Licence 1-16
1.9.4 Demonstration Version 1-16
1.9.5 Introductory Licence 1-17
1.9.6 Large-simulations Licence 1-18
1.9.7 Which Programs Require a GEMPACK Licence File 1-18

**1.10 Acknowledgments 1-19**

**2. HOW TO CARRY OUT SIMULATIONS WITH MODELS 2-21**

**2.1 An Example Simulation with Stylized Johansen 2-23**
2.1.1 Introduction to the Stylized Johansen Model 2-23
2.1.2 The Simulation 2-24

**2.2 Installing GEMPACK Software 2-25**

**2.3 Using GEMPACK: WinGEM or Unix/Command Prompt 2-25**

**2.4 WinGEM : Stylized Johansen Example Simulation 2-26**
2.4.1 Starting WinGEM 2-26
2.4.2 Preparing a Directory for Model SJ 2-26

2.4.3 Setting the Working Directory 2-27
2.4.4 Looking at the Data Directly using ViewHAR 2-27
2.4.5 TABLO-generated Program or GEMSIM? 2-28
2.4.6 The Example Simulation using a TABLO-generated Program 2-29
2.4.7 The Example Simulation using GEMSIM 2-33

**2.5 The Steps in Carrying Out a Simulation 2-37**
2.5.1 Steps using a TABLO-generated Program (Source-code GEMPACK) 2-39
2.5.2 Steps using GEMSIM 2-40

**2.6 Unix/Command Prompt : Stylized Johansen Simulation 2-43**
2.6.1 Looking at the Data Directly using SEEHAR 2-43
2.6.2 The Example Simulation using a TABLO-generated Program 2-45
2.6.3 The Example Simulation using GEMSIM 2-48
2.6.4 GEMSIM or TABLO-generated Program? 2-49

**2.7 Interpreting the Results 2-50**

**2.8 Specifying a Simulation 2-53**
2.8.1 Specifying a Simulation via a GEMPACK Command file 2-54

**2.9 The Updated Data – Another Result of the Simulation 2-57**

**2.10 Preparing Tables and Graphs for a Report 2-59**
2.10.1 Graphs 2-62

**2.11 Changing the Closure and Shocks 2-63**

**2.12 How to Carry out Several Simulations At Once 2-65**
2.12.1 Preparing an Equations File for Use by SAGEM 2-65
2.12.2 Running Several Johansen Simulations using SAGEM 2-66
2.12.3 The Results of these Johansen Simulations 2-68
2.12.4 The Command File SJLBJ.CMF for these Johansen Simulations 2-70
2.12.5 SAGEM - Advantages and Disadvantages 2-71

**2.13 How Johansen and Multi-step Solutions are Calculated 2-72**
2.13.1 The Linearized Equations of a Model 2-72
2.13.2 Johansen Solutions 2-73
2.13.3 Multi-step Simulations and Accurate Solutions of Nonlinear Equations 2-74

**2.14 GEMPACK Programs – An Overview 2-77**
2.14.1 Working with TABLO Input Files 2-77
2.14.2 Carrying out Simulations 2-77
2.14.3 Looking at, and Processing, Simulation Results 2-77
2.14.4 Working with Data 2-78
2.14.5 Environments for Modelling on Windows PCs 2-78
2.14.6 Other Programs For Special Tasks 2-78

**2.15 Different GEMPACK Files 2-79**
2.15.1 The Most Important Files 2-79
2.15.2 Files for Communication Between Programs 2-80
2.15.3 LOG Files 2-81
2.15.4 Stored-input Files 2-81
2.15.5 Files for Power Users 2-81
2.15.6 Work Files 2-81
2.15.7 Summary of Files 2-82

**2.16 For New Users - What Next? 2-83**

## 3. BUILDING OR MODIFYING MODELS 3-85

**3.1 Writing Down the Equations of a Model 3-87**
3.1.1 Writing Down the Equations of Stylized Johansen 3-87

**3.2 Data Requirements for the Linearized Equations 3-90**
3.2.1 Data Requirements for Stylized Johansen 3-90

**3.3 Constructing the TABLO Input File for a Model 3-91**
3.3.1 Viewing the TABLO Input File 3-91
3.3.2 Constructing Part of the TABLO Input File for Stylized Johansen 3-92
3.3.3 The TABLO Input File for the Stylized Johansen Model 3-95
3.3.4 Completing the TABLO Input File for Stylized Johansen 3-98
3.3.5 Change or Percentage-Change Variables 3-100
3.3.6 Variable or Parameter ? 3-101
3.3.7 TABLO Language - Syntax and Semantics 3-101

**3.4 Linearized TABLO Input Files 3-102**
3.4.1 A Linearized TABLO Input File for Stylized Johansen 3-103
3.4.2 Noteworthy Features in the Linearized TABLO Input File 3-106
3.4.3 Analysing Simulation Results 3-109
3.4.4 Writing UPDATE Statements 3-109
3.4.5 Numerical Versions of Linearized Equations 3-110
3.4.6 Numerical Examples of Update Statements 3-111
3.4.7 How the Equations Are Recalculated During Each Step of a Multi-step Calculation 3-112

**3.5 Levels TABLO Input Files 3-113**
3.5.1 Levels TABLO Input File for Stylized Johansen 3-113

**3.6 Mixed, Linearized or Levels TABLO Input Files? 3-118**

**3.7 TABLO Linearizes Levels Equations Automatically 3-118**

**3.8 Condensing Models 3-119**
3.8.1 Substituting Out Variables 3-119
3.8.2 Condensation Examples 3-120
3.8.3 Backsolving for Variables 3-124
3.8.4 Condensation Examples using Backsolving 3-124
3.8.5 Results on the Solution File in Condensation Examples 3-126
3.8.6 Should All Substitutions be Backsolves? 3-127
3.8.7 Omitting Variables 3-127
3.8.8 Condensation Actions Can Be Put on the TABLO Input File 3-128

**3.9 Creating the TABLO Input File and Command Files for Your Own Model 3-129**
3.9.1 Correcting Errors in TABLO Input Files 3-129
3.9.2 Correcting Errors in Command Files 3-131

## 4. CONSTRUCTING DATA FILES 4-135

**4.1 Header Array and Text Data Files 4-136**
4.1.1 Header Array Files 4-136
4.1.2 Text Data Files 4-136

**4.2 Constructing the Header Array Data File for Stylized Johansen 4-138**

**4.3 Modifying Data Using MODHAR 4-143**

**4.4 Using TABLO for Data Manipulation 4-146**

**4.5 Using ViewHAR to Create or Modify Header Array Files 4-149**
4.5.1 Creating a New Header Array File 4-149
4.5.2 Creating a New Header 4-149
4.5.3 Modifying Data At This Header 4-150
4.5.4 Creating a Second Header 4-150
4.5.5 Save the File 4-150
4.5.6 Adding Set and Element Labelling Information 4-151
4.5.7 Pasting from a Spreadsheet Program to ViewHAR 4-152
4.5.8 Other ViewHAR Capabilities 4-152

**4.6 Which to Use – MODHAR or ViewHAR? 4-152**

**4.7 Checking Data is Balanced 4-153**
4.7.1 SJCHK.TAB to Check Balance for Stylized Johansen 4-153
4.7.2 SJCHK.TAB to Check Balance of Updated Data 4-153
4.7.3 GTAPVIEW for Checking and Summarising GTAP Data 4-154
4.7.4 Checking the Balance of ORANIG Data 4-154

**4.8 Which Program to Use in File Conversion 4-154**

**4.9 Further Information 4-155**

## 5. COMMON FEATURES OF GEMPACK PROGRAMS 5-157

**5.1 Responding to Prompts 5-157**
5.1.1 Responding to Prompts (Upper Case or Lower Case) 5-157
5.1.2 Default Response to Questions and Prompts 5-158
5.1.3 How the Current Directory Affects Filename Responses 5-158
5.1.4 How the Current Directory Affects Filenames in Command Files 5-159
5.1.5 Directories Must Already Exist 5-159
5.1.6 Windows PC GEMPACK Programs Echo Current Directory 5-159

**5.2 Comments in Input from the Terminal 5-159**

**5.3 Interactive and Batch Operation, Stored-input and Log Files 5-160**
5.3.1 Invalid Input When Using Options 'sif' or 'asi' 5-162
5.3.2 Differences Between Batch and Interactive Program Dialogues 5-162
5.3.3 Terminal Output and Log Files 5-163
5.3.4 LOG Files and Early Errors 5-164

**5.4 Other Program Options 5-164**
5.4.1 Other Options Common to All Programs 5-164
5.4.2 Options Specific to Different Programs 5-165

**5.5 Specifying Various Files on the Command Line 5-166**
5.5.1 Command Line Option "-los" (Alternative to "-lon") 5-166
5.5.2 Command Line Option "-lic" to Specify the GEMPACK Licence 5-167
5.5.3 Abbreviations for LOG File Name on Command Line 5-167

**5.6 Fortran Compilers and Memory Management 5-168**

**5.7 Error Messages 5-169**

**5.8 GEMPACK File Types, Names and Suffixes 5-170**
5.8.1 Files with System-determined Suffixes 5-170

5.8.2 Suffixes of Other Files 5-170
5.8.3 Files - Binary or Text? 5-171
5.8.4 Why So Many Files? 5-172

**5.9 Allowed File and Directory Names 5-173**
5.9.1 Are File Names Case Sensitive ? 5-173
5.9.2 File Names on Windows PCs – Programs Made with LF90 or LF95 5-174
5.9.3 File Names on Windows PCs – Old Programs Made with F77L3 5-174
5.9.4 File Names Containing Spaces 5-175
5.9.5 Characters in Stored-input Files and Command Files 5-175

## 6. A GUIDE TO THE REST OF THE DOCUMENTATION 6-177

**6.1 Other Relevant Publications 6-178**

## 7. RECENT GEMPACK RELEASES 7-179

**7.1 Release 7.0 (October 2000) - Changes From Earlier Versions of GEMPACK 7-179**
7.1.1 GEMPACK Documentation for Release 7.0 7-184

**7.2 Release 6.0 (October 1998) and Release 6.0-001 (March 1999) 7-185**
7.2.1 New Features in Release 6.0 (October 1998) 7-185
7.2.2 New Features in Release 6.0-001 (March 1999) 7-190
7.2.3 GEMPACK Documentation for Releases 6.0 and 6.0-001 7-193

**7.3 Release 5.2 (September 1996) 7-194**
7.3.1 New Features in Release 5.2 7-194
7.3.2 GEMPACK Documentation for Release 5.2 7-195

**7.4 Earlier Releases 7-196**

## 8. REFERENCES 8-197

## 9. GEMPACK DOCUMENTS 9-199

## 10. INDEX 10-201

# CHAPTER 1

# 1. Introduction

This document is an introduction to the current version (Release 8.0, October 2002) of GEMPACK.

GEMPACK (**G**eneral **E**quilibrium **M**odelling **PACK**age) is a suite of general-purpose economic modelling software especially suitable for general and partial equilibrium models. It can handle a wide range of economic behaviour and 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 equations of the model. These equations can be written as levels equations, linearized equations or a mixture of these two.

The software provides a range of utility programs for handling the economic data base and the results of simulations, and is fully documented with plenty of examples.

GEMPACK provides

- a simple language in which to describe and document the equations of your economic model;
- a program which converts the equations of your model to a form ready for running simulations with the model;
- options for varying the choice of exogenous and endogenous variables and the variables shocked;
- utility programs to assist in managing the database on which the model is based. The data can be inspected, modified, converted to spreadsheets or moved to different machines (including those with different operating systems).

GEMPACK runs on a wide variety of computers including

- Pentium or similar PCs running Windows 95, 98, ME, NT, 2000 or XP,
- Unix machines, and
- other mainframe, mini and microcomputers with an ANSI standard Fortran 90 compiler.

GEMPACK programs run essentially unchanged on these different machines. However the commands to start them running usually differ between machines. If you have access to a computer with GEMPACK, this document supplemented by a small amount of machine- and site-specific information should enable you to carry out most routine modelling tasks with GEMPACK. For the machine-specific documentation, refer to chapter 6 to identify the document for the machine you are working on. If you are working on a multi-user machine, you may also need site-specific information from your GEMPACK Manager.

As an abbreviation, GEMPACK documents are referred to by their GPD numbers. This document is referred to as GPD-1. Chapter 9 gives the full list of all of the different GEMPACK documents and chapter 6 is a guide to the rest of the GEMPACK documentation.

This document is a practical introduction to GEMPACK for people with no prior knowledge of GEMPACK. Chapter 2 tells you how to carry out simulations with models, while chapter 3 tells you how to build or modify models. Chapter 4 describes data files and how to construct them. Chapter 5 describes common features of the different GEMPACK programs.

Chapter 1 contains a summary of changes for the current Release 8.0 from earlier versions of GEMPACK. Chapter 7 contains information about recent GEMPACK releases before the current one.

**New Users of GEMPACK**

We give detailed suggestions as to where to start for new users of GEMPACK below in section 1.5.1. We suggest that you read down to there and then follow those suggestions.

**Experienced Users of GEMPACK**

We give detailed suggestions for experienced users of GEMPACK below in section 1.5.2.

## *1.1 Documentation of GEMPACK*

GEMPACK is fully documented for users. There are 7 GEMPACK documents which are referred to by their GPD numbers.[1]

For Release 7.0 of GEMPACK the documentation was rewritten to incorporate developments for Release 5.2, Release 6.0 and Release 7.0 into the documentation. The GEMPACK documents were rearranged so that the program TABLO is described in GPD-2 and information for running simulations via GEMSIM, TABLO-generated programs and SAGEM, including GEMPACK Command file syntax, is given in GPD-3. All the utility programs including the Windows programs are described in GPD-4.

For Release 8.0 of GEMPACK, the GEMPACK documents have been fully updated to include new developments in appropriate places in the documentation. A summary of new developments in Release 8.0 is given in section 1.6 with cross references to the rest of the documentation.

If you are new to GEMPACK, chapter 2 of this document (GPD-1) is a good place to start hands-on examples. GPD-8 contains details of example models and hands-on instructions for several of these models.

Documents GPD-6 and GPD-7 contain installation instructions for different Windows PC versions of GEMPACK and material which relates to the use of GEMPACK on a PC.

A more complete guide to the GEMPACK documents is given in chapter 6. You should be familiar with this document GPD-1, especially chapters 2 and 3, before reading the other documents.

Adobe Acrobat PDF versions of the GEMPACK documentation are available on the GEMPACK CD. These PDF versions are also on the World-Wide Web at address

*http://www.monash.edu.au/policy/gpdoc.htm*

We give suggestions for reading the documentation for both new and experienced users in section 1.5 below.

---

[1] References to GEMPACK documents identify the document by GEMPACK Document (GPD) number, rather than by author or date. References are always to the version of the document which is current at the date of issue of the cross-referencing document. The GEMPACK documents referenced are listed in a separate section at the end of the References section of this document. Comments from readers on this or any of the GEMPACK documents, either pointing out errors, inaccuracies, omissions or obscurities, or making other suggestions for improvements, will be welcomed. Please address such comments to one of the authors at the Impact Project.
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".

## 1.2 Contacting the Centre of Policy Studies / Impact Project

For more information about GEMPACK, contact

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

## 1.3 GEMPACK World-Wide Web Site

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

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

This contains up-to-date information about GEMPACK, including information about different versions, prices, updates, courses and bug fixes. We encourage GEMPACK users to visit this site regularly.

In particular, this site contains a list of **Frequently Asked Questions (FAQs)** and answers at address

*http://www.monash.edu.au/policy/gp-faq.htm*

This is updated regularly. It is a supplement to the GEMPACK documentation. If you are having problems, you may find the solution there. We welcome suggestions for topics to include there.

There are also alternative GEMPACK web sites at addresses

*http://www.gempack.com* *http://www.gempack.com.au*

and you can send email to *info@gempack.com* or *support@gempack.com*

At present these alternative web sites merely point to the main GEMPACK site whose address is above, and email to these two email addresses is sent on to Ken Pearson.

## 1.4 GEMPACK-L Mailing List

GEMPACK-L is a mailing list designed to let GEMPACK users communicate amongst themselves, sharing information, tips etc. The GEMPACK developers occasionally make announcements on it (bugs, new releases etc). It replaces the GEMPACK Newsletters.

We encourage all GEMPACK users to subscribe to it. Once you have subscribed, you can send mail messages to all others on the list, and you will receive as mail any messages sent to the list. Information about subscribing and sending messages is below.

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

   **subscribe gempack-l**

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

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

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

3. To read mail sent to GEMPACK-L
   You should be able to use your usual mail utility to read GEMPACK-L mail and to send messages to it. [The address to which GEMPACK-L mail will be sent is the address from which the mail message "subscribe gempack-l" was sent.]

4. **To leave the list** at any time, send the one-line message
   **unsubscribe gempack-l**

   to the MAILSERV address **mailserv@cc.monash.edu.au**

## *1.5 A Guide to The GEMPACK Documentation*

We provide two alternative guides. Readers new to GEMPACK should read the one in section 1.5.1 while readers familiar with Release 7.0 (or a previous version) of GEMPACK should read the one in section 1.5.2.

This document contains **an index** which should make it easier for you to find the information you may need at different times in the future. The other GEMPACK documents also contain detailed indexes. The index in each document contains pointers into the document itself, and also into other GEMPACK documents.

### 1.5.1 For New GEMPACK Users – Getting Started

This document GPD-1 is the starting place if you are a new GEMPACK user.

We have built the introduction to GEMPACK around the sorts of modelling tasks you will want to do. The most important of these tasks are

- Carrying out simulations with an existing model. Looking at and analysing the results.
- Building a new model or modifying an existing model.
- As part of building a new model or modifying an existing model, you may need to build or modify the data files for the model.

We suggest that you begin with the first of these tasks (simulations). Chapter 2 tells you how to carry out simulations with existing models, and how to look at the results. We suggest that you read this in detail and (provided you have access to a computer on which GEMPACK is installed[2]) carry out the simulations described there for yourself. This chapter includes detailed hands-on instructions for using the relevant GEMPACK programs. You will find sufficient detail there to carry out all the steps involved.

The simulations in chapter 2 are based on the Stylized Johansen model, which is a small model. Even if your purpose in using GEMPACK is to work with another model (possibly ORANI or GTAP), we strongly recommend that you work through chapter 2 in detail first. After that, you will be in a position to carry out simulations with your chosen model.

At the end of chapter 2 we give suggestions as to what to do next. Roughly speaking, the possibilities are:

- If you mainly want to carry out simulations with another standard model, you will find a list of the models supplied with GEMPACK in section 1.8. You will find detailed hands-on guidance in GPD-8 about carrying out standard simulations with many of these models.
- When you are ready to build your own model (or modify someone else's), or if you just want to understand how a model is implemented in GEMPACK, you should read chapter 3. Perhaps read it quickly the first time and then go back for a more detailed study.
- If you need to build or modify the data files for a model, you should read chapter 4.
- When you want to know more about the GEMPACK utility programs (say, for report writing or post-simulation processing of results), look in GPD-4. GPD-4 contains detailed documentation about the GEMPACK utility programs.

---

[2] If you do not yet have GEMPACK available, you can download the GEMPACK Demonstration Version (see section 1.9.4) from our Web site and install it on a PC. The introductory model Stylized Johansen used in chapter 2 will run satisfactorily with the Demonstration Version.

- When you want to know more about running the GEMPACK programs, read chapter 5. This chapter gives detailed suggestions for more efficient use of the programs whether you are running them interactively or in batch mode.

We now encourage you to skip the rest of this chapter and to go straight to chapter 2.

### 1.5.2 Guide For Experienced GEMPACK Users

If you have worked with Release 7.0 (or an earlier version) of GEMPACK, the main thing you will want to see is a list of the new features. The new features in Release 8.0 are listed in section 1.6 below. If you have not used Release 7.0, you will find a list of the new features introduced in Release 7.0 and the previous releases in chapter 7.

The programs supplied with GEMPACK are the same as with Release 7.0. See section 1.7 below for a list and brief description of these programs.

The models supplied with GEMPACK are as for Release 7.0. In addition several extensions of these models incorporating explicit complementarities (for example, quotas and tariff-rate quotas) have been added. An intertemporal, rational expectations version ORANI-INT of ORANI is now available. See section 1.8 below for a list and brief description of these models.

You should also know that there are the same versions of GEMPACK (Source-code, Executable-image and Demonstration) and the same types of GEMPACK licences as there were with Release 7.0. Full details about versions and licences can be found in section 1.9 below.

We suggest that you read the list of new features in section 1.6 below. Use the cross references there to follow up those that interest you.

## *1.6 New Features in Release 8.0 of GEMPACK*

We describe the new features in Release 8.0 below.

### 1.6.1 Main New Features

1. **Complementarities** (for example, **quotas**) can be **modelled explicitly and easily** via the new COMPLEMENTARITY statement in TABLO Input files. See chapter 16 of GPD-3 for the basic ideas, and sections 3.19 and 4.14 of GPD-2 for the TABLO syntax and semantics. This is the major new feature in Release 8.0. There are several detailed examples you can work through in chapter 16 of GPD-3. See also Harrison *et al* (2002).
2. GEMPACK source code is now **limited to Fortran 90 compilers**. **Fortran 77 compilers (including Lahey F77L3) are no longer supported.** In particular, Source-code users on Windows PCs must use either LF90 or LF95. See section 5.6.
3. We have made significant changes to the documentation. In particular, we have
   (i) attempted to provide a more clearly defined path for new users (see section 1.5.1),[3]
   (ii) provided more hands-on examples for new users, mainly in GPD-1, and
   (iii) provided more substantial examples for experienced users, mainly in GPD-2, GPD-3 and GPD-4.

### 1.6.2 TABLO Input files

4. **Condensation actions can be included on the TABLO Input file** by using the new TABLO statements OMIT, SUBSTITUTE and BACKSOLVE. Originally condensation actions were only given at the Condense stage of TABLO, either interactively or on a Stored-input file. With these new TABLO statements, condensation actions can be given either in the new way on the TABLO Input file, or in the original way at the Condense stage. These TABLO condensation statements save having to look after a separate Stored-input file for condensation and help you to remember to condense large models. See sections 2.4 and 3.16-3.18 of GPD-2 for details.
5. **Elements from run-time sets** (that is, when element names are read at run-time) **can be used in TABLO Input files** (for example, in Formulas and Equations). See section 4.2.3 of GPD-2.
6. **PROD**, **MAXS** and **MINS** operators can be used in **TABLO Input files** for products and finding maximum or minimum values over a set. The syntax is similar to that for SUM. Conditions are allowed. See section 4.4.2 of GPD-2.
7. New functions **ROUND, TRUNC0** and **TRUNCB** for **rounding and truncating real numbers to integers**. ROUND rounds a real to the nearest integer. TRUNC0 truncates a real to produce the nearest integer towards zero. TRUNCB truncates a real to produce the first integer below. See section 4.4.4 of GPD-2 for details.
8. There have been some additions to **SET syntax**. As well as the usual SET Union there is now a **disjoint set union**, and sets can be set **equal to other sets**. See sections 3.1.1 and 3.1.2 of GPD-2.
9. **Set products** (sometimes called Cartesian products) are allowed via a statement of the form
   `SET  <set1> = <set2> x <set3> ;`
   The elements of set1 are of the form xx_yyy where xx ranges over all elements of set2 and yyy ranges over all elements of set3. [Essentially set1 consists of all pairs (xx,yyy).] See sections 3.1.4 and 4.6.10 of GPD-2.

[3] We are grateful to Michael Kohlhaas for suggesting this.

10. **Projection mappings from a set product** to one of the sets making the product are allowed via statements of the form
    `MAPPING (PROJECT)  <map-name> from set1 to set2 ;`
    where set1 has been defined earlier to be the product of set2 with another set. This MAPPING statement declares the mapping and sets its values. See section 3.13.2 of GPD-2.

11. New qualifiers **LINEAR_NAME=** and **LINEAR_VAR=** can be used when you are declaring a levels variable. These make it easier to **add a levels section** to a model written with linear variables and equations. See section 2.2.2 of GPD-2.

12. You can ask TABLO to automatically add **HOMOTOPY terms** to all or selected **levels equations**.
    (a) This can assist in obtaining solutions to levels equations for which the natural data is not a solution. In particular, this can be used to produce (for example, from data for one year) a non-steady-state **intertemporal data set** which satisfies all the equations of an **intertemporal model**, including the intertemporal ones (which probably would not be satisfied if the data for this one year is replicated to all years). See section 7.6 of GPD-2.
    (b) Alternatively this can be used to find (based on an initial guess) an **accurate solution to a system of levels equations**.
    This methodology can be used when only some of the equations of the model are written as levels equations in the TAB file. [There may also be many linear equations written in the TAB file.] See section 7.6 of GPD-3.

### 1.6.3 Running TABLO

13. You can use **levels names for variables during Condensation.** If you are substituting out, backsolving for, or omitting a variable which is declared as a levels variables, you can use either the levels name or the associated linear name. See sections 3.8.2, 3.8.4, 3.8.7 and 3.8.8.

14. Changes to **TABLO-generated programs**.
    TABLO-generated programs can be written in a slightly different way using new TABLO Code option **FC5.** The resulting code is **compiled more quickly by** the Lahey Fortran compiler **LF95**. For example, MMRF-GREEN compiles in about 10 minutes instead of an hour or more. See section 5.1.2 of GPD-2.

### 1.6.4 Command files for TABLO-generated Programs and GEMSIM

15. You can use **levels names for variables in Command files** for GEMSIM and TABLO-generated programs instead of the associated linear variable names. For example, in SJLB.CMF you could write
    `exogenous XFAC ;`  ! instead of "exogenous p_XFAC ;"
    `shock XFAC("labor") = 10 ;`  ! instead of "shock p_XFAC("labor") = 10 ;"
    You can use levels variable names when you are specifying the closure, the shocks, subtotals, cumulatively-retained endogenous and XAC-retained variables. [However, you still cannot use levels variable names in Command files for SAGEM.] See section 5.10 of GPD-3.

16. In certain circumstances you can specify the **desired post-simulation levels value** of a variable directly on a Command file via the new
    **final_level** <variable> **= ... ;**
    statement. Such statements are alternatives to the familiar shock statement. For example, in SJLB.CMF you could put
    `final_level  XFAC("labor") = 4.4 ;`  ! indicates post-sim levels value
    instead of

`shock p_XFAC("labor") = 10;` ! indicates percentage change
See section 5.6 of GPD-3.

17. There are two new forms of **shock statement**
**change** <variable> **= ... ;** and
**percent_change** <variable> **= ... ;**
which you can use if you know the original levels value of a variable. This can make it possible to specify a percentage change shock to a change variable via the "percent_change … ;" statement or a change shock to a percentage-change variable via the "change …. ;" statement. [These are alternatives to the usual shock statements.] See section 5.7 of GPD-3.

18. **Lines of Command files** can now be up to **500 characters long**. See section 2.7 of GPD-3.

### 1.6.5 Simulations with TABLO-generated Programs and GEMSIM

19. **Overall Accuracy Summary** when using **more than one subinterval**. If you use more than one subinterval, the Log file contains an Extraplation Accuracy Summary for the variables in each of the subintervals. The Overall Accuracy Summary combines these subinterval summaries, and shows an Extrapolation Accuracy Summary for the whole result. See section 7.3.1 of GPD-3.

20. If a **closure** specified on a Command file is **not valid**, GEMSIM or the TABLO-generated program echoes the lists of exogenous and endogenous variables. These variables are now **grouped according to argument type**. [For example, all scalar variables are first. Then all variables with a single argument ranging over the set COM. And so on.] The number of condensed equations with the same argument type is given for each type.[4] This may assist in identifying and fixing the closure problem. See section 5.2.6 of GPD-3 for more details.

21. Changes to **TABLO-generated programs**.
On Windows PCs there is no need to use "Auxiliary files = … ;" statements in Command files to help them to **locate their Auxiliary files**. See section 3.2 of GPD-3 for details.

22. When you encounter a **structurally singular LHS Matrix during the first step** of a simulation, the software offers to **solve a modified system of equations**. The solution to these modified equations may assist you (using ViewSOL and/or AnalyseGE) in identifying the cause of the singularity. See section 15.2.1 of GPD-3.

23. If you are using **automatic accuracy** and the **Left Hand Side Matrix is singular** in one step of a subinterval, the subinterval is redone with a shorter length. This may work around a singularity along the path from the pre-simulation values of the exogenous values to the post-simulation values. See section 7.4 of GPD-3.

24. When you are solving using **automatic accuracy**, it can be difficult to tell during the run how far the **simulation has progressed**. The programs produce a new output file which you can examine to check on progress. See section 7.4.7 of GPD-3.

25. At present TABLO gives no warning about possibly **uninitialised coefficients** if two or more partial initialisations are made. By default, GEMSIM and TABLO-generated programs now **initialise the values of** all parts of all partially initialised **Coefficients** to zero. You can change this. See section 6.7 of GPD-3.

26. When writing an updated Header Array file, if the long name (see section 3.1.2 of GPD-4) is blank on the header from which the data is read, the Coefficient labelling information from the TABLO Input file where the relevant Coefficient was declared is written as the **long name on the updated file**. See section 4.2.4 of GPD-3.

---

[4] SAGEM does not show this grouping.

### 1.6.6 Windows Programs

27. Many **improvements** have been made to the **Windows programs**, especially AnalyseGE, TABmate, ViewHAR and ViewSOL. See section 2.9 of GPD-4 or "What's New" in the On-line help of each of the programs WinGEM, RunGEM, AnalyseGE, ViewHAR, ViewSOL and TABmate for details.

28. In particular, **AnalyseGE** (see section 2.6 of GPD-4) has many new features. AnalyseGE can now handle, and evaluate, expressions involving conditions, functions, set mappings and integer coefficients. AnalyseGE can show useful information when you click on a Complementarity statement in the TABmate form. AnalyseGE can evaluate expressions involving the new operators PROD, MAXS and MINS (see above) and can evaluate expressions involving the new functions ROUND, TRUNC0 and TRUNCB (see above). AnalyseGE can load a CVL file (see the point below) if you wish to analyse the output from a data-manipulation TABLO Input file. AnalyseGE can load post-simulation Coefficient values (from a UDC file – see the point below) or the average of pre-simulation and post-simulation Coefficient values (from an AVC file – see the point below) as an alternative to loading the pre-simulation values from an SLC file.

29. You can write a **UDC (UpDated Coefficient values) file** and an **AVC (AVerage Coefficient values) file** when carrying out a simulation. You can load a UDC or AVC file instead of the SLC file into **AnalyseGE**. You can write a **CVL (Coefficient VaLues) file** if you are running a data-manipulation program or if you do not carry out a simulation with a model containing equations. You can load a CVL file into AnalyseGE in order to analyse the output of a data-manipulation TABLO Input file or in order to analyse why a simulation does not solve. See sections 8.5 to 8.7 of GPD-3.

30. **SLC files** now contain **set and element labelling** for all Coefficients (as do CVL, UDC and AVC files). SLC, CVL, UDC and AVC files can contain the values of **system-initiated coefficients** (produced when TABLO carries out condensation). See section 8.4 of GPD-3.

31. **Demonstration versions of RunDynam and RunMONASH are now available** from the GEMPACK web site. These come with applications based on the ORANIG-RD (a recursive, dynamic version of ORANI-G – see section 1.13 of GPD-8) and MONASH [Dixon and Rimmer (2001) and (2002)] respectively. In particular you can use the Demonstration Version of RunMONASH to replicate the results of Motor Vehicle Tariff application, as described in Dixon and Rimmer (2002). See section 2.7.1 of GPD-4 for more information.

### 1.6.7 Utility Programs

32. There are two new options in **SEEHAR**:
    SSS Short SS output
    SES Short SSE output
    These options are similar to SS and SSE but most of the **comment and blank lines are omitted**. See section 4.1 of GPD-4.

33. Programs **ACCUM** and **DEVIA** can now handle the contribution of **subtotal results** for a series of year-on-year dynamic simulations. Using the new option SUB, ACCUM gives output with each line of the cumulative solution followed by separate lines containing the subtotal contributions. See sections 10.2.4 and 10.3.3 in GPD-4 for details.

### 1.6.8 All GEMPACK Programs

34. It is possible to **abbreviate the name of the Log file on the command line**. See section 5.5.3.
35. You can ask all GEMPACK programs to **delete a specified file** either if the program **completes successfully**, or if it **stops with an error**. This might be useful when GEMPACK programs are being run from DOS BAT files or via other programs (such as Borland Delphi windows programs). See section 5.4.1.
36. On Windows PCs, all GEMPACK programs echo the **full path name of the program** and the **current** (that is, working) **directory** early on each run. See section 5.1.6.
37. Any **TAB character** in a **Command file** or in a **Stored-input file** is converted to a space. See section 5.9.5.
38. We have checked through the GEMPACK source code and the Windows programs to make sure they can always handle **long file names** and **file names with spaces**.

**Changes made for Release 7.0** (October 2000).

These are listed in section 7.1. We encourage users of Release 8.0 who have not used Release 7.0 to read these before beginning to use Release 8.0.

**New features in Release 6.0 and Release 5.2** are listed in sections 7.2 and 7.3 respectively.

## *1.7 The GEMPACK Programs*

The programs included in the current Release of GEMPACK are as follows.

1. For implementing models and carrying out simulations:

   | | |
   |---|---|
   | TABLO | for processing the description of your model |
   | GEMSIM | for carrying out simulations with your model |
   | GEMPIE | for printing simulation results |
   | SAGEM | for carrying out Johansen simulations |
   | MODHAR | for building or modifying data files |

   TABLO is documented in GPD-2, GEMSIM, SAGEM and GEMPIE are documented in GPD-3 and MODHAR is documented in GPD-4.

2. Utility programs:

   | | |
   |---|---|
   | SUMEQ | for information about the numerical equations |
   | SEEHAR,CMPHAR,CMBHAR,SUMHAR | for data management |
   | SLTOHT | for post-solution processing of simulation results |
   | SEENV | for determining the closure from Environment or Solution files |
   | ACCUM, DEVIA | for accumulating results and finding deviations |
   | TEXTBI | for extracting text files (eg, TABLO Input or Command files) from binary files |
   | CONLF | for converting Header Array files to different formats |

   These programs are documented in GPD-4.

3. Software for transferring models between different machines (that is, different operating systems):

   MKHAR, RWHAR, MKEQ, RWEQ, RWSOL, MKSOL, CMPSOL, COMPEQ

   These programs are documented in chapter 11 of GPD-4.

4. Windows programs (which are only available for PCs running Windows 95, 98, ME, NT, 2000 or XP):

   | | |
   |---|---|
   | WinGEM | Windows interface to GEMPACK |
   | GemEdit | Windows text editor sometimes used with WinGEM |
   | 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 |
   | Charter | Windows program for drawing graphs |
   | AggHAR | Windows program for aggregating data files |
   | ConvHAR | Windows program for converting Header Array files to different formats |

   These programs are documented in chapter 2 of GPD-4.

GEMPACK is suitable for models varying in size from a few equations to several thousand (or more). In particular, it contains facilities for handling very large models (such as the ORANI and MONASH models of Australia which have over a million equations).

The software consists of about 20 main programs (as listed above) and several hundred subroutines.

### 1.7.1 TABLO-Generated Programs

There is another class of programs in GEMPACK called TABLO-generated programs. These programs are not supplied as part of the GEMPACK software (on the GEMPACK CD).

TABLO-generated programs are Fortran programs **written by the program TABLO.** You need a Source-code version of GEMPACK to write TABLO-generated programs - see sections 1.9.1 and 2.4.5 below. These Fortran programs can be compiled and linked to the GEMPACK libraries of subroutines using your Fortran compiler to make an Executable image. The Executable-Image (EXE) can be used to run simulations instead of using the program GEMSIM.

## *1.8 Models Supplied with GEMPACK*

Usually at least the following models are supplied with GEMPACK.

- Stylized Johansen, a small example general equilibrium model designed as an introduction to the issues involved in building and solving such models (see Chapter 3 of Dixon *et al* (1992)),
- Miniature ORANI, a pedagogical model designed to introduce some of the essential ideas behind the ORANI model of the Australian economy (see sections 3-9 of Dixon *et al* (1982)),
- TRADMOD, a flexible multi-country trade model documented in Hertel *et al* (1992),
- ORANI-G, a comparative-static version of the ORANI model of the Australian economy,
- ORANIG-RD, a Recursive Dynamic (forecasting) version of ORANI-G, developed by Mark Horridge,
- ORANI-F, the forecasting version of the ORANI model of the Australian economy, as documented in Horridge *et al* (1993),
- GTAP, the Global Trade Analysis Project's model for analysing trade issues, as documented in Hertel (1997),
- DMR, the well-known Dervis, De Melo, Robinson model of Korea, as documented in Chapter 4 of Dixon *et al* (1992),

and four intertemporal models

- TREES, a stylized model of forestry designed to show how intertemporal models are implemented within GEMPACK, described in Codsi *et al* (1992),
- CRTS, a single sector investment model, described in Wilcoxen (1989) or Exercises 5.1-5.4 of Chapter 5 of Dixon *et al* (1992), and
- 5SECT, a 5 sector investment model designed as an introduction to the issues involved in building and solving intertemporal models, also described in Wilcoxen (1989) or Part C of Problem Set 5 of Dixon *et al* (1992).
- ORANI-INT which is a 13-sector fully intertemporal version of ORANI developed by Michael Malakellis and used for policy analysis. This model, which is fully documented in Malakellis (2000) and Malakellis (1994), is available from the GEMPACK web site. You can download the files and follow the instructions to replicate the policy simulations reported in Malakellis (2000).

New with Release 8.0 are extensions of some of the above models to include explicit complementarities (in particular, explicit quotas and tariff-rate quotas). These are

- MOIQ which is Miniature ORANI with import quotas added. See section 16.4 of GPD-3 for more details.
- G5BTRQ and G5GTRQ which are GTAP with bilateral and global tariff-rate quotas (respectively) added. See sections 16.8.4 to 16.8.6 of GPD-3 for more details.
- G94-XQ.TAB and G94-IQ.TAB which are the 1994 version of GTAP with export and import quotas (respectively) added. See sections 16.8.7 and 16.8.8 of GPD-3 for more details.

More details about all of these models (including the related files usually sent with GEMPACK) are given in chapter 1 of GPD-8.

## *1.9 Different Versions of GEMPACK and Associated Licences*

GEMPACK is sent to users either as source code or as executable images. An introduction to the different versions is given below.

Source-code versions are available on a number of different machines (see section 1.9.1 below). The other versions of GEMPACK (Limited and Unlimited Executable-image and Demonstration Versions) are only available for Pentium or similar PCs running Windows 95, 98, ME, NT, 2000 or XP.[5] A suitable Fortran compiler is required with a Source-code version, but not with the other versions of GEMPACK.

Modellers with any version of GEMPACK can carry out the full range of modelling tasks, including building and solving new models, and modifying existing ones. The only restrictions are with the Limited Executable-image version and Demonstration version, where there are limits on the size of the models that can be handled.

All versions of GEMPACK except for the Demonstration Version require a GEMPACK licence, as described below. Of the four types of licences, the most expensive is a Source-code licence and the least expensive is an Introductory licence. New licence files are required for Release 8.0 of GEMPACK.

GEMPACK licences are usually site licences, that is, multi-user licences which can be used on any number of computers at the same site within the relevant organisation.

For more information, contact us at the Centre of Policy Studies / Impact Project (see section 1.2).

Full details about GEMPACK versions, licences and current prices are available on the GEMPACK Web site at address

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

### 1.9.1 Source-code Versions and Licences

As with previous releases of GEMPACK, Source-code licences provide the most flexibility for modellers. With a Source-code version and licence, the size of the models that can be handled is limited only by the amount of memory on the computer on which the GEMPACK software is installed. With the Source-code version, a suitable Fortran compiler is required. Large models are usually solved using TABLO-generated programs. A TABLO-generated program is model specific and can solve large models considerably faster than the general-purpose program GEMSIM can.

Source-code versions are currently available for Pentium or similar PCs running Windows 95, 98, ME, NT, 2000 or XP, and for Unix machines. Other machines may be added in the future.

### 1.9.2 Limited Executable-image Version and Licence

"Limited" means that the size of models that can be solved is limited. "Executable-image" means that the software is supplied as EXE files, programs ready to run on a Windows PC (instead of the Fortran source-code which must be compiled and linked using a Fortran compiler).

Modellers with the Limited Executable-image version of GEMPACK can carry out the full range of modelling tasks, including building and solving new models, and modifying existing ones. The only

[5] In Release 7.0 we are no longer supporting Windows 3.1 since all the Windows programs supplied are now designed for the later Windows operating systems.

restrictions are on the size of the models that can be handled.[6] With this version, all simulations are carried out with the GEMPACK program GEMSIM.

With this version and licence, the size of models that can be solved is limited to what we call "medium-sized" models. For example, this version is able to solve most single-country models with up to about 40 sectors (for example, it will solve 37-sector ORANIG or 33-sector MONASH), and this version will usually solve 10-region, 10-commodity GTAP; but it will not solve 45-sector MONASH, 50-sector ORANIG or 10-region, 12-commodity GTAP. The full details of size limits for simulations with the Limited Executable-image version can be seen in section 6.2 of GPD-7.

This is the type of GEMPACK software which is usually supplied at training courses, run at the Global Trade Analysis Project at Purdue University, USA, and at the Centre of Policy Studies at Monash University.

If users with a Limited Executable-image version find that their models have become too large, they can upgrade to a Source-code version or, if they satisfy the attached conditions, to an Unlimited Executable-image version.

### 1.9.3 Unlimited Executable-image Version and Licence[7]

"Unlimited" means that the size of models that can be solved is not limited.

The Unlimited Executable-image version is only available in special circumstances, normally as a single-user licence for PhD students. It is similar to a Source-code version in that the size of models is limited only by the amount of memory on the PC on which the software is installed. With this version, all simulations are carried out with the GEMPACK program GEMSIM. With large models (for example, 115 sector MONASH or 15-region, 15-commodity GTAP), GEMSIM is noticeably slower than the corresponding TABLO-generated program (which can only be created with a Source-code licence). [Some CPU times are reported in chapter 4 of GPD-8.]

If users with an Unlimited Executable-image version find that their simulations are taking an unacceptably long time, they can upgrade to a Source-code version.

### 1.9.4 Demonstration Version

The Demonstration version is available at no cost from the GEMPACK Web site. It does not require a GEMPACK licence (or a Fortran compiler). It can only handle small models (for example, Miniature ORANI and 3-region,3-commodity GTAP). It is made available so that modellers can check out the capabilities of GEMPACK and also for teachers to use in courses introducing GE modelling.

Modellers with this version of GEMPACK can carry out the full range of modelling tasks, including building and solving new models, and modifying existing ones. The only restrictions are on the size of the models that can be handled.

This version can handle all the models in section 1.8 except for ORANI-G, ORANI-F, ORANI-INT and TRADMOD, but it can only handle the 3x3 version of GTAP.

Copies of this version can be obtained free from the Word-Wide Web (see section 1.3 for the address).

---

[6] This version of GEMPACK was called simply "the Executable-image version" for Release 5.2 of GEMPACK. There was no Unlimited Executable-image version with Release 5.2.

[7] The Unlimited Executable-image Version and licence were introduced in Release 6.0.

### 1.9.5 Introductory Licence

An Introductory GEMPACK licence may be required to run a TABLO-generated program [see (i) below] or when using some of the more recent features of the Windows programs (including ViewHAR, ViewSOL, RunGEM, RunDynam and AnalyseGE) distributed with GEMPACK [see (ii) below]. This type of licence was introduced in Release 7.0 of GEMPACK.

(i) **TABLO-generated programs**.

A GEMPACK user with a Source-code licence can create executable images of TABLO-generated programs to solve the models they build or modify. These TABLO-generated programs can be distributed to others (including others who do not have a GEMPACK licence) so that they can carry out simulations with the model.[8] TABLO-generated programs produced with Release 6.0 (or later) of GEMPACK require an Introductory licence (or a Large-simulations licence – see section 1.9.6 below) if the model they are used with is larger than medium sized (that is, exceeds the size of models which can be handled with a Limited Executable-image version – see section 1.9.2 above and section 6.2.2 of GPD-7). Thus, for example, a modeller with the TABLO-generated program for the GTAP model will usually not require a licence to solve a 10-region,10-commodity version of GTAP but will require an Introductory licence to use the same program to solve most (or all) simulations with a 10-region,12-commodity version of GTAP.

If you have a Release 7.0 Introductory licence, it will still let you run Release 8.0 TABLO-generated programs with large models.

(ii) **Windows programs (ViewHAR, AnalyseGE etc)**.

Current versions of the windows programs associated with GEMPACK (namely WinGEM, ViewHAR, ViewSOL, RunGEM, TABmate and AnalyseGE) can be downloaded from the GEMPACK web site. These programs do not require a GEMPACK licence for much of their functionality. However some of the more recent features of these programs require a moderately recent GEMPACK licence. For example,

- ViewHAR can be used to modify the data on a Header Array file. However, if the resulting file is large, you will not be able to save it without some kind of Release 6.0 (or later) GEMPACK licence.
- AnalyseGE normally requires some kind of Release 7.0 (or later) licence (unless the Solution file being analysed is very small).

The least expensive licence currently available to satisfy these licence requirements is the Introductory licence.

This licence is called an Introductory licence since it is typically needed by someone who has no version of the GEMPACK programs proper installed (except possibly the Demonstration Version) and who has obtained GEMPACK-related material (for example, a TABLO-generated program and/or some data files) from another GEMPACK user.

[8] A person using this TABLO-generated program can change the pre-simulation data, closure and shocks, but cannot change the model equations in any way. More details about distributing TABLO-generated programs to others can be found in section 3.5 of GPD-3.

### 1.9.6 Large-simulations Licence

Large-simulation licences were introduced with Release 6.0 of GEMPACK to satisfy requirement (i) in section 1.9.5 above. These licences have been replaced in Release 7.0 by Introductory licences, since the word "Introductory" covers both aspects of the licence given in the previous section.

If you have a Release 6.0 Large-simulations licence, it will still let you run Release 8.0 TABLO-generated programs with large models.

### 1.9.7 Which Programs Require a GEMPACK Licence File

New GEMPACK licences are required for Release 8.0.

You must have a suitable GEMPACK licence to run TABLO or GEMSIM.[9]

Large TABLO-generated programs require a licence in some circumstances (see section 1.9.5 above).

SAGEM requires a GEMPACK licence when used with large models (see section 6.2.3 of GPD-7). If you have any GEMPACK licence [Introductory, Limited or Unlimited Executable-image, or Source Code], there are no limits (apart from the total memory on your computer) on the size of models which can be solved using SAGEM. This applies even if you have an old or expired licence.

You do not to require a licence file for the GEMPACK utility programs.[10]

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.

---

[9] Slightly different versions of TABLO.EXE and GEMSIM.EXE are distributed with the Source-code, Executable-Image and Demonstration Versions of GEMPACK. The appropriate licence is required to run these programs.

[10] The utility programs are GEMPIE and MODHAR plus those listed under 2 and 3 in section 1.7 above.

## 1.10 Acknowledgments

We are grateful to all of those who beta tested Release 8 including Kevin Hanslow, Tom Hertel, Lars-Bo Jacobsen, Michael Kohlhaas, Markus Lips, Terry Maidment, Hans van Meijl, Chantal Nielsen, Frank van Tongeren, Wusheng Yu and our colleagues at the Centre of Policy Studies. We are grateful to those who beta tested earlier releases.

We are grateful to Michael Kohlhaas who encouraged us to provide a sharper focus for new users in our documentation.

We are grateful to Alan Powell for his support during all the years that GEMPACK has taken to develop. His positive outlook, constructive ideas and dedication to proper documentation have helped all the people who have worked on GEMPACK.

We are grateful to Peter Wilcoxen for suggesting the use of Command files and for providing us with a prototype implementation. Their use since Release 5.0 of GEMPACK has greatly facilitated the carrying out of simulations.

We are grateful to our colleague Mark Horridge who wrote ViewHAR, ViewSOL and TABmate, which are such a vital part of GEMPACK, and who introduced us to the Delphi software (used for WinGEM, RunGEM, AnalyseGE and RunDynam) for constructing Windows programs. We are grateful to Paul Harrison who wrote much of the code for WinGEM over the summer of 1995-96.

We are grateful to Tom Hertel, Robert McDougall and other colleagues at GTAP who have provided helpful suggestions and feedback on various aspects of GEMPACK over many years. We are also grateful to the many GEMPACK users who have contributed to the development of GEMPACK through suggestions and feedback.

Finally, we are indebted to Peter Dixon, Maureen Rimmer and Philip Adams and all our other colleagues at the Centre of Policy Studies at Monash University for their ideas, encouragement and support. They provide us with an ideal environment in which to develop this type of software.

## *1.11 Acknowledging the Use of GEMPACK in Reports and Papers*

When you report results obtained using GEMPACK, we ask you to acknowledge this by including a reference to GEMPACK in your paper or report. [This acknowledgement is a condition of all GEMPACK licences.]

For example, please include a sentence or footnote similar to the following.

> **The results reported here were obtained using the GEMPACK economic modelling software [Harrison and Pearson (1996)].**

The relevant paper is:

> W.J. Harrison and K.R. Pearson, "Computing Solutions for Large General Equilibrium Models Using GEMPACK", *Computational Economics*, Vol. 9 (1996), pp.83-127.

Please include this paper amongst your references.

# CHAPTER 2

# 2. How to Carry Out Simulations with Models

This is where we expect new users of GEMPACK to start learning about simulations. This chapter describes how simulations are carried out.

In this section, we explain some of the terms used in GEMPACK: implementation, simulation, levels and percentage-change variables.

In section 2.1, there is a very brief overview of the Stylized Johansen model and the simulation that you will carry out. The rest of this chapter consists of detailed instructions for carrying out the simulations, interpreting their results, and setting the simulations up.

We encourage you to work through this whole chapter before starting to work with GEMPACK on your own model. At the end of this chapter we suggest different directions you may wish to go in.

**Implementation**

A model is implemented in GEMPACK when

- the equations describing its economic behaviour are written down in an algebraic form, following a syntax described later in this document, and
- data describing one solution of the model are assembled, to be used as a starting point for simulations.

In the early versions of GEMPACK, the equations had to be written down in a linearized form, usually expressed in terms of percentage changes in the variables. Now you can choose to base your implementation on the original (usually nonlinear) equations or on a linearized representation of these. In either case you need to write them down in a text file which we call a **TABLO Input file** (since TABLO is the name of the GEMPACK program which processes this information).

The procedure for implementing models is described in detail in chapter 3.

**Simulation**

Once a model is implemented, the model can be used to carry out simulations. Many simulations are the answer to "What if" questions such as "If the government were to increase tariffs by 10 percent, how much different would the economy be in 5 years time from what it would otherwise have been?". From the original solution supplied as the starting point, a simulation calculates a new solution to the equations of the model. Within GEMPACK, the results of a simulation are usually reported as percentage changes from the original solution. Levels results may also be available for models implemented and solved with Release 6.0 (or later) of GEMPACK.

Solving models within GEMPACK is always done in the context of a simulation. You specify the values of certain of the variables (the exogenous ones) and the software calculates the values of the remaining variables (the endogenous ones).

The new values of the exogenous variables are usually given by specifying the percentage changes (increases or decreases) from their values in the original solution given as part of the implementation.

**Levels and Percentage-Change Variables**

When the model is implemented, the equations may be linearized (that is, differentiated). The variables in these linearized equations are usually interpreted as percentage changes in the original variables. The original variables (prices, quantities etc) are referred to as the **levels variables** and the (usually nonlinear) equations relating these levels variables are called the **levels equations**.

For example, the levels equation

$$D = P\,Q$$

relates the dollar value D of a commodity to its price P ($ per ton) and its quantity Q (tons). The linearized version of this is

$$p\_D = p\_P + p\_Q$$

(as explained later in chapter 3 below) which says that, to first order, the percentage change p_D in the dollar value is equal to the sum of the percentage changes p_P in the price and p_Q in the quantity.

**Data**

The data for a model often consists of input-output data (giving dollar values) and parameters (including elasticities). The data given are usually sufficient to read off an initial solution to the levels equations. (Usually all basic prices are taken as 1 in the initial solution.)

## *2.1 An Example Simulation with Stylized Johansen*

In this chapter we show you how to carry out simulations with an existing model (that is, one built by someone else). We use as an example the Stylized Johansen model described in Chapter 3 of Dixon et al (1992), hereafter referred to as **DPPW**.

The Stylized Johansen model is chosen because it is a simple, well-documented and well-known model. Once you know how to carry out simulations with it in GEMPACK, you will find it easy to carry out simulations with other, more complicated models (including ones you build yourself).

### 2.1.1 Introduction to the Stylized Johansen Model

The Stylized Johansen model is a self-contained model of a single country. The model recognises two sectors "s1" and "s2" each producing a single commodity, one household sector and two primary factors (labor and capital).

The initial input-output data base is as shown below in Table 2.1.1a, which is the same as Table E3.3.1 in DPPW (Dixon *et al* (1992)). For example, households consume 4 (million) dollars' worth of commodity 2 and industry 2 uses 3 (million) dollars' worth of labor. The amounts in the last row and column are totals.

| | | Industry | | Households | Total Sales |
|---|---|---|---|---|---|
| | | 1 | 2 | | |
| **Commodity** | 1 | 4.0 | 2.0 | 2.0 | 8.0 |
| **Sectors** | | | | | |
| **Commodity** | 2 | 2.0 | 6.0 | 4.0 | 12.0 |
| **Labor** | 3 | 1.0 | 3.0 | | 4.0 |
| **Factors** | | | | | |
| **Capital** | 4 | 1.0 | 1.0 | | 2.0 |
| **Total Production** | | 8.0 | 12.0 | 6.0 | |

**Table 2.1.1a: Input-output Data Base for Stylized Johansen**

In the GEMPACK implementation, the levels variables are as in Table 2.1.1b.

| GEMPACK variable | Meaning | DPPW Notation |
|---|---|---|
| Y | Value of household income | Y |
| PC(i) | Price of commodity i | $P_i$ (i=1,2) |
| PF(f) | Price of factor f | $P_f$ (f=3,4) |
| XCOM(i) | Supply of commodity i | $X_i$ (i=1,2) |
| XFAC(f) | Supply of factor f | $X_f$ (f=3,4) |
| XH(i) | Household use of commodity i | $X_{i0}$ (i=1,2) |
| XC(i,j) | Intermediate input of commodity i to industry j | $X_{ij}$ (i,j=1,2) |
| XF(f,j) | Input of factor f to industry j | $X_{fj}$ (f=3,4;j=1,2) |
| DVCOMIN(i,j) | Dollar values for intermediate inputs | (i,j=1,2) |
| DVFACIN(f,j) | Dollar values for factor use by industry | (f=3,4;j=1,2) |
| DVHOUS(i) | Dollar values for household consumption | (i=1,2) |

**Table 2.1.1b: Levels Variables of Stylized Johansen**

Note that most of the variables have one or more arguments (indicating associated sectors and/or factors). We refer to such variables as **vector variables.** Variables which have no arguments ('Y' is the only one here) are referred to as **scalar** or **macro variables**. GEMPACK assumes that all variables are vector variables, and treats the scalars as vectors of length 1. For example,

PC(i) is regarded as a vector variable with 2 components, one for each sector,
namely PC("s1") and PC("s2"),

XF(f,j) is regarded as a vector variable with the following 4 components:
component 1 XF("labor","s1") : input of labor (factor 1) to sector 1
component 2 XF("capital","s1") : input of capital (factor 2) to sector 1
component 3 XF("labor","s2") : input of labor (factor 1) to sector 2
component 4 XF("capital","s2") : input of capital (factor 2) to sector 2

Corresponding to each of these levels variables, there is an associated percentage change variable. TABLO adds the prefix "p_" to the name of the levels variable to indicate a percentage change. For example, p_XF is the percentage change in the levels variable XF. In DPPW, lower case letters are used to denote percentage-change variables.

More details about the model are given in chapter 3. The full TABLO Input file can be found in section 3.3.3. Full details about the model can be found in Chapter 3 of DPPW.

### 2.1.2 The Simulation

In the example simulation with the Stylized Johansen model used throughout most of this chapter, we choose the closure in which supplies of the two factors, labor and capital, are the exogenous variables. This means we will specify the percentage changes in the variable XFAC, namely p_XFAC, and solve the model to find the percentage changes in all the other variables. You will also be able to see the levels results (for example, the post-simulation value of household income), as well as the percentage change results.

For this simulation, we increase the supply of labor by 10 per cent and hold the supply of capital fixed.

The starting points for any simulation with the Stylized Johansen model are

- the TABLO Input file (called SJ.TAB on most machines) and
- the data file (called SJ.DAT on most machines).

## 2.2 Installing GEMPACK Software

**If your computer is a Windows PC**,[11] this document assumes that you have already installed the GEMPACK software on the PC. If you have a Source-Code version of GEMPACK, the instructions for installing GEMPACK are given in GEMPACK document GPD-6. If you have the Executable-Image version, these instructions are in GPD-7.

If you are working on a Windows PC, you should also install the GEMPACK Windows programs:

- WinGEM (the Windows interface to GEMPACK),
- ViewHAR (the Windows viewer for Header Array data files),
- ViewSOL (the Windows Solution viewer),
- TABmate (the Windows TABLO Input file editor), and
- AnalyseGE (for analysing simulation results)

The installation of these programs is also described in GPD-6 and GPD-7.

**If your computer is not a Windows PC**, you may be using Unix or some other operating system. This document assumes that you have already installed the GEMPACK software on your computer. Instructions for installing GEMPACK are supplied with the GEMPACK software.

## 2.3 Using GEMPACK: WinGEM or Unix/Command Prompt

There are two ways of operating GEMPACK.

- One method is working at the Command line (for example, under Unix). This method also includes working at the Command prompt under DOS or in a DOS box within Windows. We will refer to this method as **Unix/Command Prompt**.
- The other method is working on a PC in a Windows environment using GEMPACK through the WinGEM interface. We will refer to this method as **WinGEM**. If you are used to working on a PC under Windows and have WinGEM available on your computer, we suggest you use this method at least initially since WinGEM helps you with the names of the programs and files used by GEMPACK. At a later stage you may wish to work at the Command Prompt, for example, if you are running batch files for long complicated simulations.

For both methods we recommend using a working directory to keep all the example files for the Stylized Johansen model together (and to keep these files separate from the GEMPACK programs in the main GEMPACK directory).[12]

For those using the first method **Unix/Command Prompt**, skip the next section 2.4 about WinGEM, read the summary in section 2.5 and follow the instructions in section 2.6.

For those using **WinGEM**, follow the instructions in section 2.4 to start WinGEM and to copy the Stylized Johansen files to a working directory. Then carry out the examples in the rest of section 2.4 to run the example simulation with the Stylized Johansen model. After you have run the simulation, read the summary of the steps in a simulation in section 2.5.

---

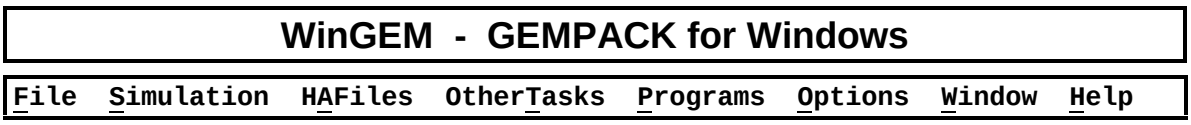
[11] By this we mean a PC (usually a Pentium) running Windows 95, 98, ME, NT, 2000 or XP.

[12] With Release 7.0 (or later) of GEMPACK, it is possible to use **long file names and directory names** provided the Fortran compiler you are using can handle them. On Windows PCs, the Lahey Fortran compilers LF90 and LF95 can handle long file names but the old Lahey Fortran compiler F77L3 can not handle more than the original 8.3 format (8 character plus a full stop and a 3 character suffix). The Executable Image version of GEMPACK can handle long file names. See section 5.9 for more details.

## *2.4 WinGEM : Stylized Johansen Example Simulation*

### 2.4.1 Starting WinGEM

In Windows, double click on the ***WinGEM*** icon to start GEMPACK for Windows. This should give the main WinGEM menu, as shown below, across the top of the screen. [You may need to look closely to see this since WinGEM is rather self-effacing and only occupies a small part of the top of you screen - the rest of the screen is as it was before you double-clicked on WinGEM.]




### 2.4.2 Preparing a Directory for Model SJ

To keep all example files for the Stylized Johansen model together in one area, we show you how to create a separate directory **\SJ** for these files and how to copy the relevant files into this directory. This can be done by copying the files using Explorer (or My Computer) in a windows way, or by using the DOS *copy* command.

**(a)** To copy these files within Windows,

change to Explorer in Windows by the usual Windows method. Use Explorer to create a **new** folder or subdirectory called **\sj** and copy all the **sj*.*** files from the directory containing the GEMPACK model examples (usually C:\GP\EXAMPLES) to this directory **\sj**. Return to WinGEM to continue these examples.

**(b)** Alternatively, to use DOS to copy the files,

click on ***File*** in the main WinGEM menu. Select

***Shell to DOS***

to get to the DOS prompt. Then you can use the DOS Command

**md \sj**

to create this directory. To change to this directory and copy all the SJ*.* files from the directory containing the GEMPACK model examples (usually C:\GP\EXAMPLES), you can use the commands (you will need to change the second of these if your examples are not in C:\GP\EXAMPLES):

**cd \sj**

**copy c:\gp\examples\sj*.***

**dir**

This should list about 20 files associated with the Stylized Johansen model.

Return from the DOS box to WinGEM by typing **exit**

### 2.4.3 Setting the Working Directory

WinGEM uses the idea of a working directory to simplify choosing files and running programs. This working directory is where all the files for the model you are using are stored.

For the Stylized Johansen model examples here, the working directory needs to be the directory \SJ you have just created. To set this, first click on ***File*** in the main WinGEM menu. This will produce a drop-down menu. In the drop-down menu, click on the menu item

***Change both default directories...***

The notation we use for the sequence of clicks (first *File* then *Change both default directories)* is

***File | Change both default directories...***

In the file selection box that appears, choose drive **C:** (or the drive containing your directory \SJ if it is on a different drive). Then double-click on **C:** (this will be at the top of the list of directories shown) and then double-click on the subdirectory **SJ**. [Make sure that the directory name shown in blue above the selection box changes to C:\SJ (or D:\SJ etc if your \SJ directory is on another drive).] Click on the ***Ok*** button.

(This is similar to the DOS commands **C:** and **cd \sj** .)

### 2.4.4 Looking at the Data Directly using ViewHAR

The input-output data used in the Stylized Johansen model are contained in the data file SJ.DAT. This is a special GEMPACK *binary* file - called a **Header Array file** - so you cannot just look at it in a text editor. Instead you will look at SJ.DAT using one of the programs for reading Header Array files, called ViewHAR. Select from the main WinGEM menu :

***HA Files | View VIEWHAR***

The ViewHAR window will appear. Click on ***File | Open...*** and select the file SJ.DAT.

This will open the file SJ.DAT and show its contents on the Contents screen.

Each of the rows corresponds to a different array of data on the file. Look at the column under the heading Name to see what data are in these arrays.

| | Header | Type | Size | Name |
|---|---|---|---|---|
| **1** | **CINP** | **RE** | **SECTxSECT** | **Intermediate inputs of commodities to ind..** |
| 2 | FINP | RE | SECTxFACT | Intermediate inputs of primary factors - dollar |
| 3 | HCON | RE | SECT | Household use of commodities - dollar values |

The first array is the "Intermediate inputs of commodities to industries - dollar values".
The Header **CINP** is just a label for this array. (Headers can have up to 4 characters.)
The array is of Type **RE.** The **R** means this is an array of real numbers. The **E** means that this array has set and element labelling (see chapter 5 of GPD-4).

Double click on **CINP** to see the numbers in this array.

| DVCOMIN | s1 | s2 | Total |
|---|---|---|---|
| s1 | 4.000000 | 2.000000 | 6.000000 |
| s2 | 2.000000 | 6.000000 | 8.000000 |
| Total | 6.000000 | 8.000000 | 14.000000 |

Compare these numbers with the input-output data for Stylized Johansen shown in Table 2.1.1a. The actual data in the file at this header is just the 2x2 matrix. ViewHAR calculates and shows the row and column totals.

To return to the Contents Screen, click on ***Contents*** in the ViewHAR menu.

Look at the other Header Arrays called **FINP** and **HCON** to see where their numbers fit in the input-output data base.

Close ViewHAR in the normal Windows way by selecting ***File*** | ***Exit*** .

### 2.4.5 TABLO-generated Program or GEMSIM?

If you have the Source-code version of GEMPACK, you will usually carry out simulations using a TABLO-generated program; the steps in this case are spelled out in section 2.4.6 below. Alternatively, you could also use the GEMSIM method (as spelled out in section 2.4.7 below) so you might like to learn about this method also. Note that, except with small models, you will usually find that using a TABLO-generated program is quicker.

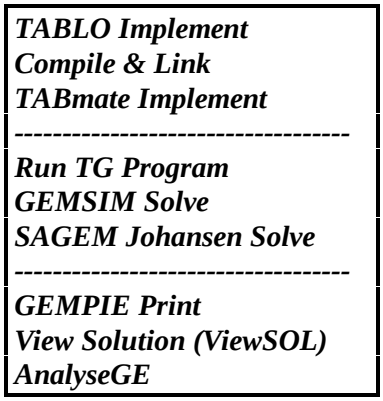
If you have an Executable-image version of GEMPACK, or the Demonstration version of GEMPACK, you must carry out simulations using the program GEMSIM; the steps in this case are spelled out in section 2.4.7 below.

In either case we illustrate the steps for the Stylized Johansen simulation described in section 2.1.2 above. Of course the simulation results are the same whichever method you use. If you don't have the Source-code version, please skip section 2.4.6 and go straight to section 2.4.7.

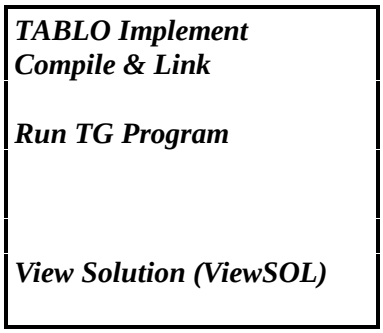
### 2.4.6 The Example Simulation using a TABLO-generated Program

In the next examples we are assuming that you have the Source-code version of GEMPACK and also have access to a Lahey Fortran compiler (either LF90, LF95) which should be on your DOS PATH.

From the WinGEM menu at the top of the screen choose ***Simulation.*** In the drop-down menu the choices are




The items from this menu you will be using in this simulation are




In the TABLO-generated program method, the GEMPACK program TABLO is used to convert the algebraic equations of the economic model into a Fortran program specific to your model. This Fortran program (which is referred to as the **TABLO-generated program** or **TG Program** in the above menu) is compiled and linked to a library of GEMPACK subroutines. The executable image of the TABLO-generated program produced by the compiler is used to run simulations on the model (instead of using the program GEMSIM). This method provides faster execution times for large models than the alternative of using GEMSIM (see section 2.4.7) but means you must have an appropriate Fortran compiler.

There are three steps involved in carrying out a simulation using GEMPACK.

Step 1 - Implement the model

Step 2 - Solve the equations of the model

Step 3 - View the results

WinGEM will guide you through these steps and indicate what to do next.

**Step 1 - Implementing the model SJ using TABLO**

**Step 1(a) - Run TABLO to create the TABLO-generated program**

The TABLO Input file is called **SJ.TAB**. It contains the theory of the Stylized Johansen model. Choose

***Simulation | TABLO Implement...***

A window for TABLO will appear. Click on the ***Select*** button to select the name of the TABLO Input file **SJ.TAB**.[13] This is all TABLO needs to implement the model.[14]

In the menu for the TABLO window, select ***Options*** menu item. Then in this menu choose

***TABLO Options...***

A new TABLO Options window will appear giving many and varied special options for TABLO. Ignore all options for now except the two in the top left hand corner which read

```
O   PGS  Generate GEMSIM auxiliary files
O   WFP  Generate Fortran code
```

There is a black dot in one of the buttons. Click on the second option **WFP** because we want you to create the TABLO-generated Fortran program. Then click on the ***Ok*** button to return to the TABLO window.

Click on the ***Run*** button. The program runs TABLO in a DOS box and when complete, returns you to the TABLO window with the names of files it has created: the Information file SJ.INF and the Log file. Look at both of these files by clicking the ***View*** buttons beside them.

The Information file SJ.INF gives information about the TABLO Input file such as whether there are any syntax or semantic errors found by the program TABLO when it was checking the TABLO Input file. Search the file for %% to see if there are any errors. Search the file for "syntax error" to see how many syntax errors and semantic problems there are (hopefully none). Go to the end of the file to see what actions can be carried out by the TABLO-generated program produced in this TABLO run.

**Step 1(b) - Compile and Link the TABLO-generated Program**

When you have looked at these two files, click on the ***Go to Compile and Link*** button at the bottom of the TABLO window to run the Fortran compiler. (Alternatively you can start this window by choosing *Simulation | Compile and Link...* from WinGEM's main menu.)

In the Compile and Link window, the file SJ.FOR is already selected as the TG Program Name. Click on the button ***Compile and Link*** and wait a little while the compiler converts the Fortran file SJ.FOR into the executable image SJ.EXE.

When finished, click on the button ***Go to 'Run TG Program'*** to proceed to the next step in running a simulation: Step 2 - Solve the equations of the model.

---

[13] This TABLO Input file is shown in full in section 3.3.3 below. For the present, we suggest that you take this on trust and continue working through Step 1.

[14] It is possible to carry out all of Step 1 (including Compile and Link) from within TABmate by selecting instead

***Simulation | TABmate Implement***

**Step 2 - Solve the equations of the model using TABLO-generated program**

The button takes you to the window for running the TABLO-generated program SJ.EXE. (Alternatively you can start this window by choosing *Simulation | Run TG Program...* from WinGEM's main menu.)

First ***Select*** the Command file called SJLB.CMF. Since Command files are text files, look at this Command file in the text editor by clicking the ***Edit*** button.

How is the closure specified? What shock is applied?

What data file is used by this model? How many steps are used in the multi-step solution?

[Details about using a GEMPACK Command file to specify a simulation are given in section 2.8 later in this document. For the present, we suggest that you take this on trust and continue with the simulation.]

Then select ***File | Exit*** to return to the "Run TG Program" window.

Click on ***Run*** to run SJ.EXE with the Command file SJLB.CMF.

If SJ.EXE produces the Solution file, click on ***Go to ViewSOL***.[15]

If there is an error, view the Log file.

There is no point in trying to look at the Solution file SJLB.SL4 directly in a text editor because it is a binary file, not a text file. ViewSOL will open the Solution file and display it on the screen.

**Step 3 – View the Solution using ViewSOL**

The *Go to ViewSOL* button starts the program ViewSOL running and opens the Solution file SJLB.SL4 . [Alternatively you can start this window by choosing *Simulation | View Solution (ViewSOL)...* from WinGEM's main menu.]

The Solution file is already selected.

You will see the **Contents** page listing many of the variables of the model. ViewSOL has 3 slightly different formats for this Contents list. Select ***Format...*** from ViewSOL's main menu and there click on ***Arrange vectors by name*** (in the panel headed Vector options); then click ***Ok*** which will put you back to the Contents list.

To see the results of one of these variables listed by name, just **double-click** on the corresponding row in the Contents list. First double-click on the **p_XCOM** row to see the results for this variable (demand for the two commodities). Select 3 decimal places (see the third drop-down list box along the top row of the current ViewSOL window - the only one with a single figure in it). Then you should see something like the following:[16]

| p_XCOM | sjlb | Pre sjlb | Post sjlb | Chng sjlb |
|---|---|---|---|---|
| s1 | 5.885 | 8.000 | 8.471 | 0.471 |
| s2 | 6.899 | 12.000 | 12.828 | 0.828 |

Across the **s1** row you see the percentage change result (5.885%), the pre-simulation levels value (8.000), the post-simulation levels value (8.471) and the change (0.471); these are the results for the total supply of commodity **s1**.

---

[15] The alternative ***Go to GEMPIE*** is described a little later in the text. The option ***Go to AnalyseGE*** runs the program AnalyseGE (see section 2.6 of GPD-4) to assist you in analysing the results. A hands-on introduction to AnalyseGE can be found in chapter 6 of GPD-8.

[16] You will only see the levels results if you are using Release 6.0 of GEMPACK or later.

Then click on ***Contents*** to return to the Contents list.

To see the **p_XFAC** results, double-click on this row. You will see

| p_XFAC | sjlb | Pre sjlb | Post sjlb | Chng sjlb |
|---|---|---|---|---|
| labor | 10.000 | 4.000 | 4.400 | 0.400 |
| capital | 0 | 2.000 | 2.000 | 0 |

This time all the numbers are in red which is used to remind you that, for this simulation, both components of this variable p_XFAC are exogenous. You should easily be able to understand all of these results.

Then click on ***Contents*** to return to the Contents list.

To see the **p_XC(i,j)** results [intermediate inputs of commodity i into industry j], double-click on this row. Now you see

| p_XC | s1 | s2 |
|---|---|---|
| s1 | 5.885 | 5.885 |
| s2 | 6.899 | 6.899 |

and, in the second drop-down box you should see "**1 sjlb**". This indicates that you are just seeing the linearized simulation results (the percentage changes in the four components of this variable). You can't see the pre- and post-simulation levels results at the same time since this variable p_XC is a matrix variable. To see the pre-simulation levels results, click on the second drop-down list box (the one showing "1 sjlb") and select the second alternative ("2 Pre sjlb"). Then you will see the pre-simulation levels results. You might also like to look at the post-simulation levels results and the changes.

Then click on ***Contents*** to return to the Contents list.

When you have finished looking at the results, exit from ViewSOL. This will return you to the screen for the simulation and you can use another method of looking at the results by using the program GEMPIE.

**Step 3 (alternative) – View the results using GEMPIE**

Click on the *Go to GEMPIE* button in the TABLO-generated program window; this will take you to a GEMPIE window. [Alternatively you can start this window by choosing *Simulation* | *GEMPIE Print...* from WinGEM's main menu.]

The Solution file SJLB.SL4 is already selected. Click on ***solutions?*** to see what solutions are available for printing. Choose the Totals and levels solutions. Click the ***Ok*** button to continue.

Click on ***Run*** to run the program GEMPIE. Check that a GEMPIE Print file (called SJLB.PI5) has been created.

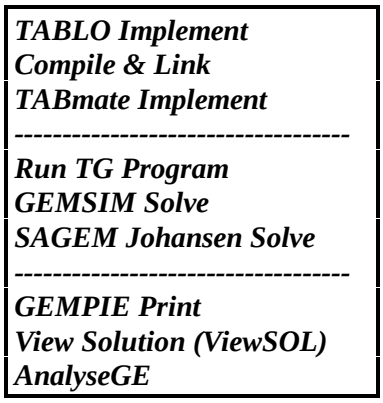
To see your results, examine the results in file SJLB.PI5 using ***View File***. What are the percentage changes in the prices of the two factors, labor and capital? [We show in detail how to read these results in section 2.7 below.]
(***File | Exit*** to return to the GEMPIE window.)

This completes the simulation. We suggest that you close all open windows (the Run TG Program, Compile & Link, and TABLO windows) by selecting ***File | Exit*** in each one. But leave WinGEM running.

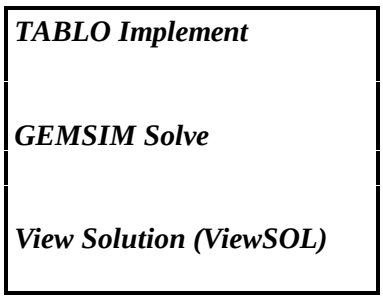
Now you can either jump to section 2.5, or you can read section 2.4.7 if you also wish to know about using GEMSIM for simulations.

### 2.4.7 The Example Simulation using GEMSIM

In the WinGEM menu at the top of the screen choose ***Simulation***. In the drop-down menu the choices are[17]




The items from this menu you will be using in this simulation are




TABLO, GEMSIM and ViewSOL are the names of programs used to carry out the three steps of a simulation :

Step 1 - Implement the model

Step 2 - Solve the equations of the model

Step 3 - View the results

WinGEM will guide you through these steps and indicate what to do next.

[17] Unless you have the Source-code version of GEMPACK, the option "Compile & Link" will probably be grey to indicate it is not available - it requires a suitable Fortran compiler.

**Step 1 - Implementing the model SJ using TABLO (for GEMSIM output)**

The TABLO Input file is called **SJ.TAB**. It contains the theory of the Stylized Johansen model. Choose

***Simulation | TABLO Implement...***

A window for TABLO will appear. Click on the ***Select*** button to select the name of the TABLO Input file **SJ.TAB**.[18] This is all TABLO needs to implement the model.[19]

In the menu for the TABLO window, select the ***Options*** menu item. Then in this menu choose

***TABLO Options...***

A new TABLO Options window will appear giving many and varied special options for TABLO. Ignore all options for now except the two in the top left hand corner which read

```
O   PGS  Generate GEMSIM auxiliary files
O   WFP  Generate Fortran code
```

There is a black dot in one of the buttons. Click on the first option **PGS** because we want you to create the GEMSIM Auxiliary files. Then click on the ***Ok*** button to return to the TABLO window.

By "implement" we mean convert the TABLO Input file into binary computer files which are used by the simulation program GEMSIM in the next step. These files are referred to as Auxiliary files (or sometimes as the GEMSIM Statement and Table files) and in this case, are called SJ.GSS and SJ.GST.

Click on the ***Run*** button. The program TABLO runs in a DOS box and when complete returns you to the TABLO window with the names of files it has created: the Information file SJ.INF and the Log file. Look at both of these files by clicking the ***View*** buttons beside them.

The Information file SJ.INF gives information about the TABLO Input file such as whether there are any syntax or semantic errors found by the program TABLO when it was checking the TABLO Input file. Search the file for %% to see if there are any errors. Search the file for "syntax error" to see how many syntax errors and semantic problems there are (hopefully none). Go to the end of the file to see what actions GEMSIM can carry out with the Auxiliary files produced in this TABLO run.

When you have looked at these two files, click on the ***Go to GEMSIM*** button at the bottom of the TABLO window to go on to the next step in running a simulation: Step 2 - Solve the equations of the model.

**Step 2 - Solve the equations of the model using GEMSIM**

The *Go To GEMSIM* button takes you to the GEMSIM window. (Alternatively you can start this window by choosing *Simulation | GEMSIM Solve* from WinGEM's main menu.)

First ***Select*** the Command file called SJLB.CMF. Since Command files are text files, look at this Command file in the text editor by clicking the ***Edit*** button.

How is the closure specified? What shock is applied?

---

[18] This TABLO Input file is shown in full in section 3.3.3 below. For the present, we suggest that you take this on trust and continue working through Step 1.

[19] It is possible to carry out Step 1 from within TABmate by selecting instead
***Simulation | TABmate Implement***

What data file is used by this model? How many steps are used in the multi-step solution?

[Details about using a GEMPACK Command file to specify a simulation are given in section 2.8 later in this document. For the present, we suggest that you take this on trust and continue with the simulation.]

Then select ***File | Exit*** to return to the GEMSIM window.

Click on ***Run*** to run GEMSIM with the Command file SJLB.CMF.

If GEMSIM produces the Solution file, click on ***Go to ViewSOL***.[20]

If there is an error, view the Log file.

There is no point in trying to look at the Solution file SJLB.SL4 directly in a text editor because it is a binary file, not a text file. ViewSOL will open the Solution file and display it on the screen.

**Step 3 - View the Solution using ViewSOL**

The *Go to ViewSOL* button starts the program ViewSOL running and opens the Solution file SJLB.SL4 .

(Alternatively you can start this window by choosing *Simulation | View Solution (ViewSOL)...* from WinGEM's main menu.)

The Solution file is already selected.

You will see the **Contents** page listing many of the variables of the model. ViewSOL has 3 slightly different formats for this Contents list. Select ***Format...*** from ViewSOL's main menu and there click on ***Arrange vectors by name*** (in the panel headed Vector options); then click ***Ok*** which will put you back to the Contents list.

To see the results of one of these variables listed by name, just **double-click** on the corresponding row in the Contents list. First double-click on the **p_XCOM** row to see the results for this variable (demand for the two commodities). Select 3 decimal places (see the third drop-down list box along the top row of the current ViewSOL window - the only one with a single figure in it). Then you should see something like the following:[21]

| p_XCOM | sjlb | Pre sjlb | Post sjlb | Chng sjlb |
|---|---|---|---|---|
| s1 | 5.885 | 8.000 | 8.471 | 0.471 |
| s2 | 6.899 | 12.000 | 12.828 | 0.828 |

Across the **s1** row you see the percentage change result (5.885%), the pre-simulation levels value (8.000), the post-simulation levels value (8.471) and the change (0.471); these are the results for the total supply of commodity **s1**.

Then click on ***Contents*** to return to the Contents list.

To see the **p_XFAC** results, double-click on this row. You will see

| p_XFAC | sjlb | Pre sjlb | Post sjlb | Chng sjlb |
|---|---|---|---|---|
| labor | 10.000 | 4.000 | 4.400 | 0.400 |
| capital | 0 | 2.000 | 2.000 | 0 |

[20] The alternative ***Go to GEMPIE*** is described a little later in the text. The option ***Go to AnalyseGE*** runs the program AnalyseGE (see section 2.6 of GPD-4) to assist you in analysing the results. A hands-on introduction to AnalyseGE can be found in chapter 6 of GPD-8.

[21] You will only see the levels results if you are using Release 6.0 of GEMPACK or later.

This time all the numbers are in red which is used to remind you that, for this simulation, both components of this variable p_XFAC are exogenous. You should easily be able to understand all of these results.

Then click on ***Contents*** to return to the Contents list.

To see the **p_XC(i,j)** results [intermediate inputs of commodity i into industry j], double-click on this row. Now you see

| p_XC | s1 | s2 |
|---|---|---|
| s1 | 5.885 | 5.885 |
| s2 | 6.899 | 6.899 |

and, in the second drop-down box you should see "**1 sjlb**". This indicates that you are just seeing the linearized simulation results (the percentage changes in the four components of this variable). You can't see the pre- and post-simulation levels results at the same time since this variable p_XC is a matrix variable. To see the pre-simulation levels results, click on the second drop-down list box (the one showing "1 sjlb") and select the second alternative ("2 Pre sjlb"). Then you will see the pre-simulation levels results. You might also like to look at the post-simulation levels results and the changes.

Then click on ***Contents*** to return to the Contents list.

When you have finished looking at the results, exit from ViewSOL. This will return you to the screen for the simulation and you can use another method of looking at the results by using the program GEMPIE.

**Step 3 (alternative) - View the results using GEMPIE**

Click on the *Go to GEMPIE* button in the TABLO-generated program window; this will take you to a GEMPIE window.

(Alternatively you can start this window by choosing *Simulation* | *GEMPIE Print...* from WinGEM's main menu.)

The Solution file SJLB.SL4 is already selected. Click on ***solutions?*** to see what solutions are available for printing. Choose the Totals and levels solutions. Click the ***Ok*** button to continue.

Click on ***Run*** to run the program GEMPIE. Check that a GEMPIE Print file (called SJLB.PI5) has been created.

To see your results, examine the results in file SJLB.PI5 using ***View File***. What are the percentage changes in the prices of the two factors, labor and capital? [We show in detail how to read these results in section 2.7 below.]
(***File | Exit*** to return to the GEMPIE window.)

This completes the simulation. We suggest that you close the windows for GEMPIE, GEMSIM, TABLO by choosing ***File | Exit*** in each separate window except the main WinGEM one. (Alternatively you can choose *File* | *Exit* in the main WinGEM window which will shut down all windows at once and exit from WinGEM.)

## *2.5 The Steps in Carrying Out a Simulation*

This section is a summary of the steps used to implement a model and carry out a simulation.

If you are working on a Windows PC, you have already carried out these steps in section 2.4. WinGEM guided you between the steps. You should treat this section as an overview of the steps and procedures you have carried out.

If you are going to follow the Unix/Command prompt method, you will carry out these steps in section 2.6 below. We put this section first for you to give you an overview of the steps you will have to carry out.

In all cases, after the TABLO Input file for a model has been written, there are 3 steps on the computer to carry out a first simulation with the model. These steps are

- Step 1. Computer implementation of the model
- Step 2. Simulation (Solve the equations of the model)
- Step 3. View the results

Step 1 always involves running the program **TABLO**.

The details of Step 2 are a little different depending on whether you have a Source-code or Executable-image version of GEMPACK, as we explain below.

Step 3 involves running either the program **ViewSOL** (Windows PCs only) or the program **GEMPIE** (which is available on all machines).

**Source-code version of GEMPACK (Windows PCs or Unix/Command prompt)**

If you have a Source-code version of GEMPACK, you will usually carry out simulations using a TABLO-generated program. In this case, the three steps are described in section 2.5.1, and these steps are illustrated in Figure 2.5.1.

**Executable-image or Demonstration version of GEMPACK (Windows PCs only)**

If you have an Executable-image version of GEMPACK, or the Demonstration version of GEMPACK, you must carry out simulations using the program GEMSIM. In this case, the three steps are described in section 2.5.2, and these steps are illustrated in Figure 2.5.2.

Users with a Source-code version of GEMPACK can also use the GEMSIM method. Except with small models, they will usually find that using a TABLO-generated program is quicker.

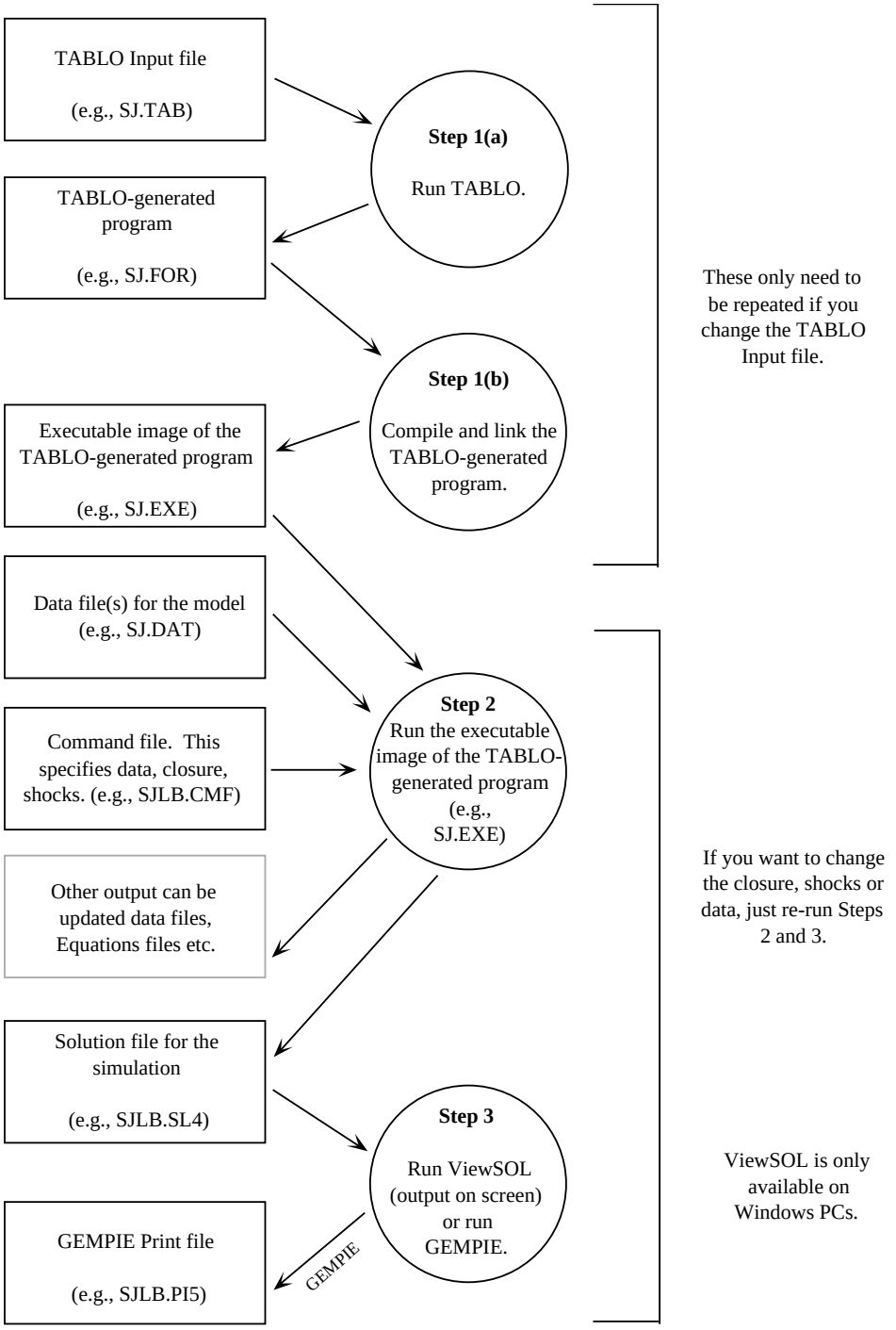



**Figure 2.5.1 The Steps in Carrying Out a Simulation using a TABLO-generated Program**

### 2.5.1 Steps using a TABLO-generated Program (Source-code GEMPACK)

These steps are illustrated in Figure 2.5.1.

**Step 1. Computer Implementation of the Model**

**Step 1(a) - Run TABLO to create TABLO-generated program**

Process the TABLO Input file for the model by running the program TABLO. Select the option WFP which tells TABLO to write a Fortran program (referred to as the **TABLO-generated program of the model**) which captures the theory of the model.

**Step 1(b) - Compile and Link the TABLO-generated program**

Compile and link the TABLO-generated program of the model, produced in Step 1(a). This will produce an **executable image** of the TABLO-generated program.[22]

**Step 2. Simulation (Solve the equations of the model)**

Run the executable image of the TABLO-generated program, as produced in Step 1(b). Take inputs from a **Command file** which tells the program which base data files are to be read and describes the closure (that is, which variables are exogenous and which are endogenous) and the shocks. This program then computes the solution to your simulation and writes the results to a **Solution file**.

**Step 3. Printing or Viewing the Results of the Simulation**

If you are working on a Windows PC, the usual way of viewing the simulation results is to run the Windows program **ViewSOL**. This lets you examine the results interactively on the screen. [Alternatives are to use the program GEMPIE or the Windows program AnalyseGE.[23]]

If you are working in the Unix/Command prompt way, you will use the GEMPACK program GEMPIE to look at the results.

If you choose to use the program GEMPIE, this converts the Solution file produced in Step 2 to a GEMPIE Print file. This file can be printed (or edited).

On all machines, another alternative is to use the program SLTOHT to process results (for example, to produce tables for reports), as introduced in section 2.10 below.

**Other Simulations**

Once you have carried out one simulation with a model, you will probably want to carry out others, for example, to change the closure and/or shocks, or even to run from different base data. In such cases, you do not have to repeat Steps 1(a) and 1(b). All you have to do is carry out Steps 2 and 3. A hands-on example is given in section 2.11 below. (Of course Steps 1(a) and 1(b) must be repeated if you change the TABLO Input file for the model in any way.)

---

[22] For more details about compiling and linking, see section 1.2 of GPD-2 and section 6.3.1 in GPD-6.

[23] A hands-on introduction to AnalyseGE can be found in chapter 6 of GPD-8.

### 2.5.2 Steps using GEMSIM

This is an alternative method used for implementing a model and carrying out simulations which uses the program GEMSIM instead of writing a TABLO-generated program. These steps are illustrated in Figure 2.5.2.

**Step 1. Computer Implementation of the Model**

Process the TABLO Input file for the model by running the GEMPACK program TABLO. Select the option PGS which asks TABLO to produce the **GEMSIM Auxiliary files** for the GEMPACK program GEMSIM (see Step 2). These files capture the theory of the model, as written in the TABLO Input file. Selecting option PGS rather than the option WFP as in section 2.5.1 above is what initiates the GEMSIM route rather than the TABLO-generated program route.

**Step 2. Simulation (Solve the equations of the model)**

Run the GEMPACK program GEMSIM [24] and tell it to use the GEMSIM Auxiliary files produced in Step 1. Take inputs from a **Command file** which tells the program which base data files are to be read and describes the closure (that is, which variables are exogenous and which are endogenous) and the shocks. GEMSIM then computes the solution to your simulation and writes the results to a **Solution file**.

**Step 3. Printing or Viewing the Results of the Simulation**

If you are working on a Windows PC, the usual way of viewing the simulation results is to run the Windows program **ViewSOL**. This lets you examine the results interactively on the screen. [Alternatives are to use the program GEMPIE or the Windows program AnalyseGE.[25]]

If you are working in the Unix/Command prompt way, you will use the GEMPACK program GEMPIE to look at the results.

If you choose to use the program GEMPIE, this converts the Solution file produced in Step 2 to a GEMPIE Print file. This file can be printed (or edited).

On all machines, another alternative is to use the program SLTOHT to process results (for example, to produce tables for reports), as introduced in section 2.10 below.

Note that Step 1 above is very similar to Step 1(a) in the TABLO-generated program case (see section 2.5.1 above). Step 1(b) in section 2.5.1 has no analogue in the GEMSIM case since GEMSIM is a general-purpose program which can be used to solve any model. Step 2 is different only in that GEMSIM is run rather than the TABLO-generated program. Step 3 is identical in the two cases.

**Other Simulations**

Once you have carried out one simulation with a model, you will probably want to carry out others, for example, to change the closure and/or shocks, or even to run from different base data. In such cases, you do not have to repeat Steps 1. All you have to do is carry out Steps 2 and 3. A hands-on example is given in section 2.11 below. (Of course Step 1 must be repeated if you change the TABLO Input file for the model in any way.)

---

[24] GEMSIM is an abbreviation for **G**eneral **E**quilibrium **M**odel **SIM**ulator.

[25] A hands-on introduction to AnalyseGE can be found in chapter 6 of GPD-8.

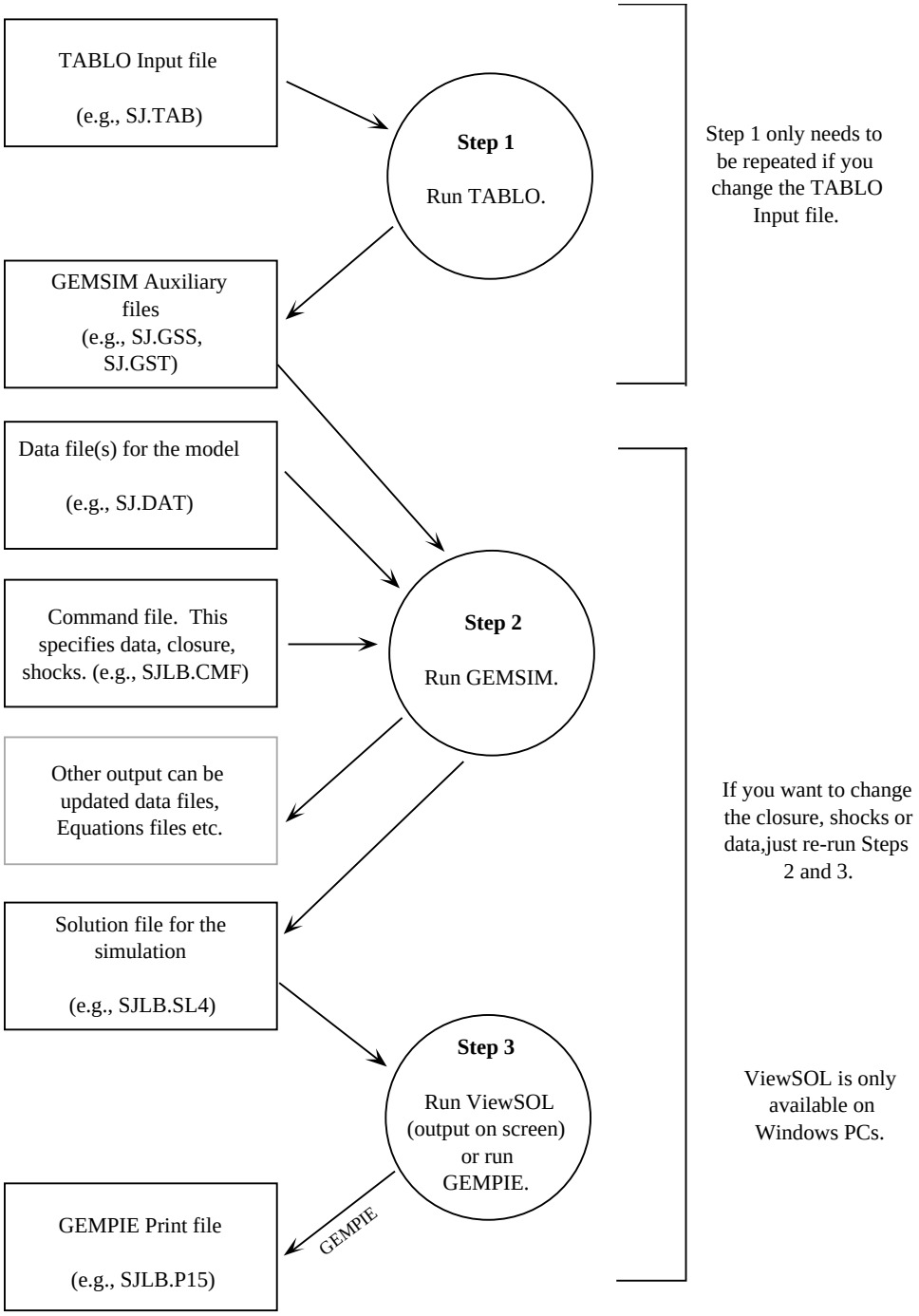



**Figure 2.5.2 The Steps in Carrying Out a Simulation using GEMSIM**

**Where to Next?**

If you are working on a Windows PC, we suggest that you now move ahead to section 2.7 "Interpreting the Results". [If you are interested in working on the PC at the Command prompt (the next section 2.6), we suggest that you return to this section 2.6 below only after you have worked through the rest of this chapter once via WinGEM.]

If you are working on a Unix computer or prefer to work at the Command prompt, please move on to the next section 2.6 which gives detailed instructions.

## *2.6 Unix/Command Prompt : Stylized Johansen Simulation*

This section is for those using the **Unix/Command Prompt** method of running GEMPACK. First create a working directory called **sj**. Then locate the GEMPACK Examples subdirectory on your computer (perhaps gp/examples or similar) and copy all the **sj*.*** files in the Examples subdirectory to the new subdirectory **sj**.

Below we show you how to carry out the Stylized Johansen simulation in section 2.1.2 on your computer.

In this section we are assuming that you have a Source-code version of GEMPACK on your computer (together with a suitable Fortran compiler). As indicated in section 2.5 above, you have the choice of using a TABLO-generated program (usually preferred since it is usually faster) or of using the program GEMSIM. We explain the TABLO-generated program method in section 2.6.2 and the GEMSIM alternative in section 2.6.3.

### 2.6.1 Looking at the Data Directly using SEEHAR

You may want to print out the data in the data files of a model. The program SEEHAR can be used to prepare a print file of all the data on a Header Array file.

Make sure you are in the working directory **sj** for the Stylized Johansen model. The command to run a program varies from computer to computer. Under many operating systems, just type the name of the program:

```
seehar
```

(or whatever command is required on your machine to run SEEHAR. If this does not work, consult your machine-specific documentation.) Then give the following user input for running SEEHAR.

> (The first, third and fourth responses are carriage-returns or Enter. After each response, we have given a comment, which starts with an exclamation mark '!'. When running the program, you should not type in the exclamation mark or the following comment - just type in the text shown in bold type.)

```
<carriage-return>    ! Use the default options
sj.dat               ! Name of the data file
<carriage-return>    ! Use the default output file name SJ.SEE
<carriage-return>    ! (as instructed by SEEHAR)
r                    ! Output the remaining (ie all) arrays
```

**User Input to SEEHAR**

This should produce the file SJ.SEE containing all the data in SJ.DAT. You can look at this file in a text editor to see the actual data.

Find the data at header **CINP** in this file, which should look like the box shown below.

```
Page 1 of Display of (all of) 'DVCOMIN(SECT,SECT)'.
This coefficient is of size 2x2.
(This data is at header 'CINP' on the file. Long name is
 'Intermediate inputs of commodities to industries - dollar values'.)
  ------------------------------------------------------------

Next submatrix contains numbers from DVCOMIN(SECT,SECT) part, which are
  positions (1-2, 1-2) of DVCOMIN(SECT,SECT).

  COLUMN                1           2         ROW
  ROW                                         TOTALS
 No. Name             s1          s2
   1 s1             4.000000    2.000000    6.000000
   2 s2             2.000000    6.000000    8.000000
   COLUMN
   TOTALS           6.000000    8.000000   14.000000
```

DVCOMIN is the name used for this data in the TABLO Input file for the Stylized Johansen model.

Compare these numbers with the input-output data for Stylized Johansen shown in Table 2.1.1a. The actual data in the file at this header is just the 2x2 matrix. SEEHAR calculates and shows the row and column totals.

Find the DVFACIN data at header FINP and find out how much labor is used by sector s2.

Find the DVHOUS data at header HCON and check where the numbers at this header fit into the input-output data base shown in Table 2.1.1a.

### 2.6.2 The Example Simulation using a TABLO-generated Program

**Step 1(a) - Run TABLO to create the TABLO-generated program**

Make sure you are in the working directory **sj** for the Stylized Johansen model. The command to run a program varies from computer to computer. Under many operating systems, just type the name of the program:

```
tablo
```

(or whatever command is required on your machine to run TABLO. If this does not work, consult your machine-specific documentation.) Then give the following user input for running TABLO.

(The first and third are carriage-returns or Enter. After each response, we have given a comment, which starts with an exclamation mark '!'. When running the program, you should not type in the exclamation mark or the following comment - just type in the text shown in bold type.)

```
<carriage-return>    ! Use the default options
sj                   ! Name of the TABLO Input file
<carriage-return>    ! Use the default Information file name

          (TABLO will take a minute or two to check that the
          formulas, equations and updates contain no errors.
          When finished a menu will appear asking what to do next.)

a                    ! Begin automatic code generation

          (The code generation menu appears.)

wfp                  ! Write a Fortran (the TABLO-generated) program
<carriage-return>    ! Use the other default code generation options
sj                   ! Name of the program to be written

           (TABLO will take a minute or two to write the code
           for the TABLO-generated program.)
```

**User Input to TABLO**

When this is finished, you can check that the following new files have been created.

- The TABLO-generated program (called **sj.f** on Unix machines and often called SJ.FOR on other machines).
- The Information file (often called **sj.inf**),[26]
- Auxiliary Statement and Table files (usually called **sj.axs** and **sj.axt** respectively) for the TABLO-generated program. These are binary files containing data for the TABLO-generated program. (They should not be deleted, renamed or moved.)

  You can examine the Information file. It should indicate that there were no syntax or semantic errors during the CHECK stage and, at the end of the CODE stage, tell you how many lines of code are in the TABLO-generated program.

[26] On Unix machines, file names are case sensitive. Usually the names of the files in this section will all be in lower case if you are working on a Unix machine.

**Step 1(b) - Compile and Link the TABLO-generated Program**

The exact command for doing this varies from machine to machine. On most machines you can simply type [27]

```
ltg sj
```

to do this (the first letter above is "L" not one). (Consult your machine-specific documentation if this does not work.) It should create an executable image (often called SJ.EXE but called **sj** on Unix machines).

**Step 2 - Solve the equations of the model using the TABLO-generated Program**

This is where the actual simulation is run. The exact command to start this running varies from machine to machine. On most machines you can simply enter

```
sj
```

At the first screen of options, choose **cmf** to use a GEMPACK Command file to run this simulation.
Enter the 2 responses

```
cmf
sjlb.cmf
```

(These responses are the same on all machines.)

Then there will be a lot of screen output. First this does a 1-step Euler solution and updates the data. Then the program calculates a 2-step solution and finally a 4-step one. Finally the extrapolated solution is calculated from these 3 solutions and the updated data based on this is calculated and written. This will take a minute or two.

In the above run, the TABLO-generated program **sj** takes all of its input from the **GEMPACK Command file sjlb.cmf**. This file, which is shown in full in Figure 2.8.1 below, contains statements containing all the necessary information for carrying out this simulation. For example, the statement

```
shock p_xfac("labor") = 10 ;
```

(don't type this) in the Command file **sjlb.cmf** tells the program **sj** to give a 10% increase to the supply of labor. We postpone a discussion of the other statements in the Command file until section 2.8.1 below.

When the program is finished, you can check that the Solution file **sjlb.sl4** has been created.[28] This contains the numerical results of the simulation.

---

[27] This is the only step that requires a suitable Fortran compiler and, normally, a Source-code version of GEMPACK.

[28] The name of the Solution file is inferred from the name sjlb.cmf of the Command file. The Solution file has the same name except that the suffix is changed from .cmf to .sl4. See section 2.8.1 below for more details.

**Step 3 - Run GEMPIE to Look at the Results**

The Solution file produced in Step 2 is a binary file which cannot be viewed or printed directly. To convert it to a form suitable for viewing on your terminal or printing, you need to run the program GEMPIE, which can usually be done by entering:

```
gempie
```

Enter the following responses when prompted by the program. As before, don't type in the exclamation marks and parts of lines after them. (In particular, if a line below begins with an exclamation mark, it is just a comment, and does not need to be entered when you run the program.)

```
<carriage-return>       ! Use the default program options
sjlb                    ! Name of Solution file (produced in Step 2)
L                       ! Use the "Lists" option to choose
!                           which endogenous variables to print
a p_y p_pc p_xh p_xf p_dvhous  ! Choose all components of these
f                       ! Finished giving list
<carriage-return>       ! Accept default name (usually sjlb.pi5)
!                           for the Print file
Test Results            ! Page heading
4                       ! Number of decimal places
```

**User Input to GEMPIE**

Check that a GEMPIE Print file (usually called **sjlb.pi5**) has been created. To check your results, view this file in your editor (or print it). [The end of this file is shown in Table 2.7 below.] For example, what are the percentage changes in the prices of the two factors, labor and capital? [We show in detail how to read these results in section 2.7 below.]

If you want to print out all the results on the Solution file instead of choosing a List as in the example above, start the program

```
gempie
```

and enter the responses:

```
<carriage-return>       ! Use the default program options
sjlb                    ! Name of Solution file (produced in Step 2)
a                       ! Print all the results
<carriage-return>       ! Accept default name (usually sjlb.pi5)
Test Results            ! Page heading
4                       ! Number of decimal places
```

**User Input to GEMPIE for all Variables**

### 2.6.3 The Example Simulation using GEMSIM

Here we spell out the GEMSIM alternative. Of course the simulation results are the same whichever method you use.

**Step 1 - Run TABLO for GEMSIM output**

If you wish to run TABLO to produce GEMSIM Auxiliary files instead of producing the TABLO-generated program, start TABLO running as before

```
tablo
```

then give the following User Input. (The one change from the User Input in section 2.6.2 is the response **pgs** instead of **wfp** ).

```
<carriage-return>    ! Use the default options
sj                   ! Name of the TABLO Input file
<carriage-return>    ! Use the default Information file name

          (TABLO will take a minute or two to check that the
          formulas, equations and updates contain no errors.
          When finished a menu will appear asking what to do next.)

a                    ! Begin automatic code generation

          (The code generation menu appears.)

pgs                  ! Produce output for GEMSIM
<carriage-return>    ! Use the other default code generation options
sj                   ! Name of GEMSIM Auxiliary files to be output

           (TABLO will take a minute or two to write the
            output required by GEMSIM.)
```

**User Input to TABLO**

When this is finished, you can check that the following new files have been created.

- The GEMSIM Auxiliary Statement and Table files (usually called **sj.gss** and **sj.gst** respectively). They are different from the Auxiliary files produced if, as in section 2.6.2 above, you select option WFP instead of PGS; the different suffixes (usually **.axs, .axt** in the TABLO-generated case and **.gss, .gst** in the GEMSIM case) indicate this.
- The Information file (often called **sj.inf**). This contains information about the run of TABLO.

  If the TABLO Input file contains errors, these will be clearly marked and explained in this Information file.

  You can examine the Information file. It should indicate that there were no syntax or semantic errors during the CHECK stage and, at the end of the CODE stage, tell you the names of the GEMSIM Auxiliary Statement and Table files.

**Step 2 - Solve the equations of the model using GEMSIM**

Enter the command to start the program GEMSIM running, (on most machines)

```
gemsim
```

When GEMSIM prompts you, enter the 2 responses

```
cmf
sjlb.cmf
```

(These responses are the same on all machines.)

Then there will be a lot of screen output. First this does a 1-step Euler solution and updates the data. Then the program calculates a 2-step solution and finally a 4-step one. Finally the extrapolated solution is calculated from these 3 solutions and the updated data based on this is calculated and written. This will take a minute or two.

In the above run, GEMSIM takes all of its input from the **GEMPACK Command file sjlb.cmf**. This file, which is shown in full in Figure 2.8.1 below, contains statements containing all the necessary information for carrying out this simulation. For example, the statement

```
shock p_xfac("labor") = 10 ;
```

(don't type this) in the Command file **sjlb.cmf** tells the program **sj** to give a 10% increase to the supply of labor. We postpone a discussion of the other statements in the Command file until section 2.8.1 below.

When the program is finished, you can check that the Solution file **sjlb.sl4** has been created.[29] This contains the numerical results of the simulation.

**Step 3 - View the Results using GEMPIE**

When GEMSIM has finished running, run GEMPIE exactly as in Step 3 in section 2.6.2 above to produce the GEMPIE Print file **sjlb.pi5**.

### 2.6.4 GEMSIM or TABLO-generated Program?

As you have seen, you have the choice of using GEMSIM or the TABLO-generated program. For small models such as Stylized Johansen, Miniature ORANI or 3-region, 3-commodity GTAP, GEMSIM is quite fast. TABLO-generated programs only give their great advantage with large models. Some CPU times are reported in chapter 4 of GPD-8.

[29] The name of the Solution file is inferred from the name sjlb.cmf of the Command file. The Solution file has the same name except that the suffix is changed from .cmf to .sl4. See section 2.8.1 below for more details.

## *2.7 Interpreting the Results*

If you are working on a Windows PC, you have looked at the results via ViewSOL (see Step 3 in section 2.4.6 or 2.4.7 above). In particular you can click on the ViewSOL Contents page and look at variables such as Macros, p_XH, p_PC, p_XF, p_DVHOUS. We have copied some of the results from ViewSOL to a spreadsheet (via the Copy menu item in ViewSOL). These are shown in Table 2.7a.

Whichever machine you are working on, the results of the simulation, as given by GEMPIE (see Step 3 (alternative) in section 2.4.6 or 2.4.7, or Step 3 in section 2.6.2 or 2.6.3), are shown in Table 2.7b on the next page. [Table 2.7b is actually a selected part of the Print file produced by GEMPIE.]

We think that you will find the tables fairly easy to interpret.

**Table 2.7a: Results (ViewSOL) from the Solution file SJLB.SL4**

| Macros | sjlb | Pre sjlb | Post sjlb | Chng sjlb |
|---|---|---|---|---|
| p_Y | 5.8853 | 6.0000 | 6.3531 | 0.3531 |
| | | | | |
| p_XH | sjlb | Pre sjlb | Post sjlb | Chng sjlb |
| s1 | 5.8853 | 2.0000 | 2.1177 | 0.1177 |
| s2 | 6.8993 | 4.0000 | 4.2760 | 0.2760 |
| | | | | |
| p_PC | sjlb | Pre sjlb | Post sjlb | Chng sjlb |
| s1 | 0.0000 | 1.0000 | 1.0000 | 0.0000 |
| s2 | -0.9486 | 1.0000 | 0.9905 | -0.0095 |
| | | | | |
| p_XF | s1 | s2 | | |
| labor | 10.0000 | 10.0000 | | |
| capital | 0.0000 | 0.0000 | | |
| | | | | |
| p_DVHOUS | sjlb | Pre sjlb | Post sjlb | Chng sjlb |
| s1 | 5.8853 | 2.0000 | 2.1177 | 0.1177 |
| s2 | 5.8853 | 4.0000 | 4.2354 | 0.2354 |

The results show what happens if the supply of labor is increased by 10 per cent and the supply of capital is held fixed. For example,

1. Look at the simulation result for 'p_Y', the percentage change in the levels variable 'Y'. The dollar value of total nominal household expenditure will increase by 5.8853 per cent from its pre-simulation value of 6.0000 to its post-simulation value of 6.3531.
2. Look at the results for p_XH. The result for commodity 2, p_XH ("s2"), shows that households will consume 6.8993 per cent more of commodity 2 than they did previously.
3. Look at the results for p_PC. The price of commodity 2 will fall by 0.9486 per cent.
4. The simulation results for p_DVHOUS show that the dollar value of household consumption of commodity 2 will rise by 5.8853 per cent from its pre-simulation value of 4.0000 to its post-simulation value of 4.2354.

**Table 2.7b: Results in GEMPIE Print File SJLB.PI5**

```
 When levels values are available for a variable,
  they are shown underneath the percent-change or change result.
 The 4 results are shown in the order:
  Percent-change (or change), Pre-simulation, Post-simulation,
Change.

 For example
      3.000 (percent change)
      500.0 (pre-sim level)
      515.0 (post-sim level)
       15.0 (change)


  PAGE 1      Test results


 p_Y   Total nominal household expenditure
      5.8853
      6.0000
      6.3531
      0.3531

 p_PC (SECT)  Price of commodity i
     s1           s2
      0.0000*     -0.9486
      1.0000       1.0000
      1.0000       0.9905
      0.0000*     -0.0095

 p_XH (SECT)  Household demand for commodity i
     s1           s2
      5.8853       6.8993
      2.0000       4.0000
      2.1177       4.2760
      0.1177       0.2760

 p_XF (FAC,SECT)  Factor inputs to industry j
  p_XF(-,s1) results where '-' is in set 'FAC'.
    labor       capital
     10.0000       0.0000
      1.0000       1.0000
      1.1000       1.0000
      0.1000       0.0000*

  p_XF(-,s2) results where '-' is in set 'FAC'.
    labor       capital
     10.0000       0.0000
      3.0000       1.0000
      3.3000       1.0000
      0.3000       0.0000*

 p_DVHOUS (SECT)  Dollar value of household use of commodity i
     s1           s2
      5.8853       5.8853
      2.0000       4.0000
      2.1177       4.2354
      0.1177       0.2354
```

Recall that, within GEMPACK, all simulations are set up and solved as perturbations from an initial solution, and results are usually reported as changes or percentage changes from this original solution. In this case the original solution values are as shown in Table 2.1.1a above, which shows million dollar values of activity. Suitable levels values for quantities can be obtained by assuming that, initially,

all prices are 1.

(This just sets the units in which quantities are measured.) Then, for example, since households consume 4 million dollars' worth of commodity 2, this means that they consume 4 million units of that commodity.

Hence the three simulation results mentioned above mean that, once labor is increased by 10 per cent and capital is held fixed,

1. total nominal household expenditure Y has increased to approximately 6.353 million dollars (5.8853 per cent more than the original value of 6 million dollars).
   (The other three values given with p_Y in Table 2.7a or Table 2.7b are
   6.0000 which is the pre-simulation level of Y,
   6.3531 which is the post-simulation level of Y and
   0.3531 the change between these two values.)
2. household consumption (XH) of commodity 2 has increased to 4.2760 million units (6.8993 per cent more than the original 4 million units),
3. the commodity price (PC) of commodity 2 has fallen from one dollar per unit to approximately 99.051 cents per unit (a fall of 0.9486 per cent), and
4. the dollar value of household consumption (DVHOUS) of the commodity produced by sector "s2" has risen from 4 million dollars to approximately 4.2354 million dollars (an increase of 5.8853 per cent).

The updated values in (2), (3) and (4) above should be related since dollar value should equal price times quantity. The levels equation for commodity 2 ("s2") is

DVHOUS("s2") = PC("s2) x XH("s2")

Note that this equation is true for the post-simulation values, since, from (2) and (3) above, the post-simulation price times the post-simulation quantity is

0.99051 x 4.2760 = 4.2354

which is equal to the post-simulation dollar value in (4). This confirms that the solution shown in the ViewSOL results or the GEMPIE Print file satisfies the levels equation connecting price, quantity and dollar value of household consumption of this commodity. You might like to check some of the other levels equations in this way.

## *2.8 Specifying a Simulation*

In order to specify the details for carrying out a simulation, you must

- say which model to use,
- say which base data to begin from (that is, the pre-simulation solution),
- say which closure (that is, which variables are endogenous and which are exogenous), and
- say which variables to shock, and by how much, and
- specify the names of the various output files.

(All of this information is shown schematically in Figure 2.8.)

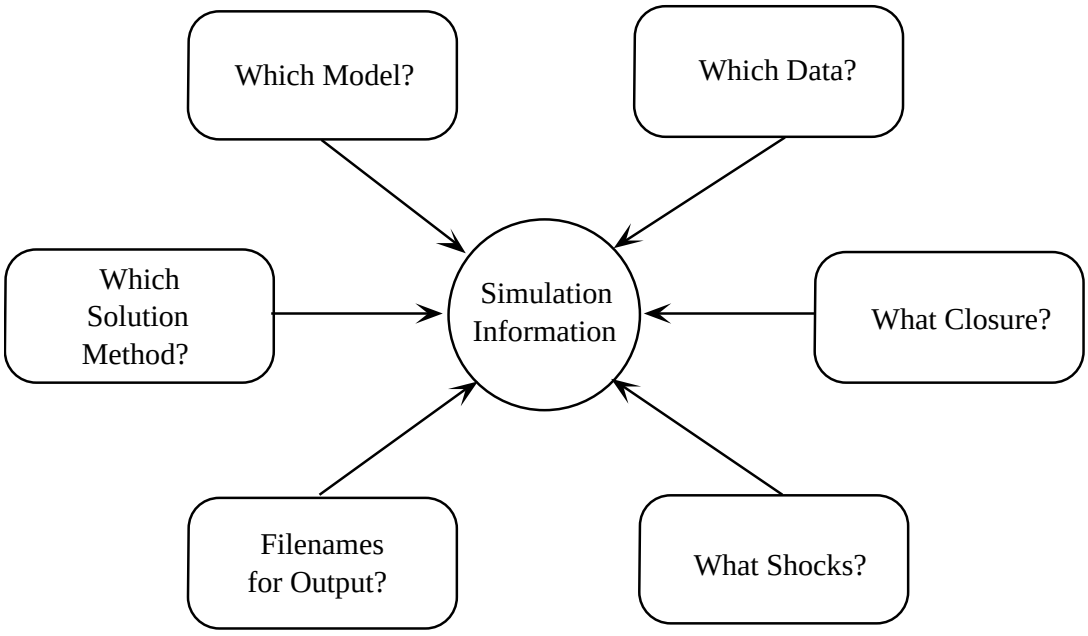



**Figure 2.8 The Information Required to Specify a Simulation**

Within GEMPACK, the normal way of specifying this information to the software is via a **Command file**. Indeed, when you carried out the example simulation above, this is exactly what happened in Step 2 above since there you ran either the TABLO-generated program or GEMSIM and took inputs from the Command file SJLB.CMF.

The instructions in this Command file must be prepared in advance in a text editor.

We explain in section 2.8.1 the statements in this GEMPACK Command file SJLB.CMF.

### 2.8.1 Specifying a Simulation via a GEMPACK Command file

In Step 2 of the example simulation, the program took all the information required to specify the simulation from the GEMPACK Command file SJLB.CMF. The file SJLB.CMF is shown in full in Figure 2.8.1.

The statements in SJLB.CMF are discussed briefly below.

The statement

```
auxiliary files = sj ;
```

tells GEMSIM or the TABLO-generated program to use the Auxiliary files produced in Step 1. (This effectively tells which TABLO Input file (or model) to work with, since these files are just a processed version of the TABLO Input file SJ.TAB for the Stylized Johansen model.) [If you are using the TABLO-generated program SJ.EXE, the Auxiliary files are SJ.AXS and SJ.AXT produced when you ran TABLO in Step 1. If you are using GEMSIM, the Auxiliary files are SJ.GSS and SJ.GST produced when you ran TABLO in Step 1.][30]

**Figure 2.8.1: The GEMPACK Command File SJLB.CMF**

```
! The following GEMPACK Command file (usually called SJLB.CMF)
!  carries out a multi-step simulation
!  for the Stylized Johansen model.

!  Auxiliary files (usually tells which TAB file)
auxiliary files = sj ;

! Data files
file iodata = sj.dat ;
updated file iodata = <cmf>.upd ;

! Closure
exogenous p_xfac ;
rest endogenous ;

! Solution method information
method = euler ;
steps = 1 2 4 ;

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

shock  p_xfac("labor") = 10 ;

verbal description =
Stylized Johansen model. Standard data and closure.
10 per cent increase in amount of labor.
(Capital remains unchanged.) ;

! Options
extrapolation accuracy file = yes ;
log file = yes ;

! End of Command file
```

[30] We realise that new GEMPACK users usually find these "auxiliary files" statements somewhat mysterious. A fuller explanation of their purpose can be found in chapter 3 of GPD-3.

The statement

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

tells SJ.EXE or GEMSIM to read base data from the file SJ.DAT (which contains the data in Table 2.1.1a above).

The line

```
! Data files
```

above this line is a comment since it begins with an exclamation mark **!**. While such comments are ignored by the software, they are very important in organising and documenting the Command file and in making it an intelligible record of the simulation. [You can see several other comment lines in the file.]

The statements

```
exogenous p_xfac ;
rest endogenous ;
```

give the closure (that is, which variables to take as exogenous and which to take as endogenous), while the statement

```
shock p_xfac("labor") = 10 ;
```

gives the shock needed to increase the supply of labor by 10 per cent.

When the TABLO-generated program SJ or GEMSIM carries out a simulation, as well as being able to report the changes in the endogenous variables, the program produces an updated version of the original data file(s). The data in these updated data files represent post-simulation values (that is, the ones that would hold after the shocks have worked their way through the economy). For Stylized Johansen, this contains post-simulation dollar values of the entries in Table 2.1.1a above. The statement

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

names the file to contain this updated data. The <cmf> in this line indicates that this part of the name comes from the name of the Command file. Since the Command file is usually called SJLB.CMF, the program replaces <cmf> by SJLB (the name of the Command file ignoring its suffix .CMF) so that the updated iodata file will be called SJLB.UPD. [You will get an opportunity to examine the updated data for this simulation in section 2.9 below.][31]

The most important output from a simulation is the Solution file which contains the results for the percentage changes in prices and quantities. Here we have omitted the name of the Solution file so the name of this file is taken from the name of the Command file. Because the Command file is called SJLB.CMF, the Solution file will be called SJLB.SL4 (the same basic name SJLB followed by .SL4 which is the standard GEMPACK suffix for Solution files). Another alternative is to add a statement of the form

***solution file = ... ;  ! Not included in this SJLB.CMF***

[31] The statement "***updated file iodata = sjlb.upd ;***" would be accepted in the Command file. The reasons for preferring to have the name of the updated data file inferred from the name of the Command file are set out in section 2.5 of GPD-3.

in the Command file. Such a statement is allowed, but it is customary to omit it so that the name of the Solution file is inferred from the name of the Command file. [32]

You are required to give a **verbal description** of the simulation. This description, which can be several lines of text, goes on the Solution file and is transferred to the GEMPIE Print file when you run GEMPIE (as in Step 3 above). You can use this to remind yourself (and others) about salient features of the simulation. The statement

```
verbal description =
Stylized Johansen model. Standard data and closure.
10 per cent increase in amount of labor.
    (Capital remains unchanged.)
1,2,4-step solutions plus extrapolation. ;
```

in SJLB.CMF give 4 lines of text for the verbal description in this case.
(The semicolon '**;**' indicates the end of this description. Note that all statements in GEMPACK Command files must end with a semicolon '**;**'.)

With GEMPACK, you can choose one of 4 related solution methods for each simulation. These are introduced in section 2.13.3 below. The statements

```
method = euler ;
steps = 1 2 4 ;
```

in the Command file tell the program to use Euler's method based on 3 separate solutions using 1, 2 and 4 steps respectively. (See section 2.13.3 below for an explanation about step numbers.)

The accuracy of the solution depends on the solution method and the numbers of steps. SJ.EXE or GEMSIM can be asked to provide information about the accuracy on an **Extrapolation Accuracy file**. The statement

```
extrapolation accuracy file = yes ;
```

asks the program to produce such a file. The information on this file is described in section 2.13.3 below. (The name of this file is the same as that of the Solution file except that it has a different suffix, namely '.XAC', which makes the full name SJLB.XAC.)

The statement

```
log file = yes ;
```

asks the software to produce a LOG file showing all the screen activity as the program runs. This LOG file is called SJLB.LOG - it takes its name from the name SJLB.CMF of the Command file but changes the suffix from .CMF to .LOG. This LOG file is a useful record of the simulation.

Further details of Command files and running simulations are given in GPD-3 *Simulation Reference: GEMSIM, TABLO-generated Programs and SAGEM.* A summary of the statements that can be used in a GEMPACK Command file for running TABLO-generated programs or GEMSIM is given in chapter 18 of the GEMPACK document GPD-3.

---

[32] The reasons for preferring to have the name of the Solution file inferred from the name of the Command file are set out in section 2.5 of GPD-3. Note that a statement "***solution file = sjlb ;***" would be allowed and that the alternative statement "***solution file = <cmf> ;***" would have the same effect since the <cmf> is replaced by the base name of the Command file (that is, with the suffix .CMF omitted).

In particular, we know that new users of GEMPACK find the auxiliary files (the statement above is "auxiliary files = sj ;") and the file statements (the statements above are "file iodata = sj.dat; " and "updated file iodata = <cmf>.upd ;") confusing initially. More details about these can be found in chapter 3 and chapter 4 respectively of GPD-3.

Note that complete documentation of Command files is given in GPD-3.

## *2.9 The Updated Data – Another Result of the Simulation*

Once the simulation shocks have worked their way through the economy, prices and quantities change (as reported by the simulation results). These price and quantity changes imply changes in the values contained in the original data base for the model. When you carry out a simulation, the GEMPACK programs compute these new values. These new values are written to a file which is referred to as the **updated data** or **post-simulation data**.

As you saw in section 2.8 above, the line

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

in the Command file SJLB.CMF means that, when you ran the simulation, the software produced the **updated data file SJLB.UPD**. This file contains the data as it would be after the shocks (in this case, the increase in labor supply) have worked their way through the model.

GEMPACK provides easy ways of looking at data files. For example, see section 2.4.4 which tells you how to use ViewHAR, or section 2.6.1 how to use SEEHAR to look at the base (or pre-simulation) data file SJ.DAT. This file is the starting point for the simulation; and, when you look at this file you see the numbers shown in Table 2.1.1a above.

You can use the same software and methods to look at the updated data in file SJLB.UPD.

**If you are using WinGEM**, select from the main WinGEM menu :

> ***HA Files | View VIEWHAR***
>
> The ViewHAR window will appear. Click on ***File | Open...*** and select the file SJLB.UPD.
>
> Each of the rows corresponds to a different array of data on the file. Look at the column under the heading Name to see what data are in these arrays. Look at the three arrays.

**If you are using SEEHAR at the Unix/Command prompt**, type in the program name

> ```
> seehar
> ```
>
> and give the following inputs:

```
<carriage-return>    ! Use the default options
sjlb.upd             ! Name of the updated data file
<carriage-return>    ! Use the default output file name SJLB.SEE
<carriage-return>    ! (as instructed by SEEHAR)
r                    ! Output the remaining (ie all) arrays
```

**User Input to SEEHAR**

> This should produce the file **sjlb.see** containing all the data in **sjlb.upd**. You can look at this file in a text editor to see the actual data.

If you do this you will see the values in the Table 2.9 (which are shown to 3 decimal places). As in Table 2.1.1a, the figures are dollar values (not quantities).

| | | Industry | | Households | Total Sales |
|---|---|---|---|---|---|
| | | 1 | 2 | | |
| **Commodity** | 1 | 4.235 | 2.118 | 2.118 | 8.471 |
| **Sectors** | | | | | |
| **Commodity** | 2 | 2.118 | 6.353 | 4.235 | 12.707 |
| **Labor** | 3 | 1.059 | 3.177 | | 4.235 |
| **Factors** | | | | | |
| **Capital** | 4 | 1.059 | 1.059 | | 2.118 |
| **Total Production** | | 8.471 | 12.707 | 6.353 | |

**Table 2.9: Updated Input-output Data in File SJLB.UPD**

You can check that these post-simulation values are consistent with the results of the simulation as discussed in section 2.7 above. For example, the p_DVHOUS results in Table 2.7a show that the value of household expenditure on commodity s2 increased by 5.8853 percent from its pre-simulation value of 4 to its post-simulation value of 4.2354 (which agrees with the Commodity 2 Households value in the table above).

The most obvious results of a simulation are the percentage changes in the variables. The updated data (which is always obtained when you run a simulation) is another important "result" of the simulation, one which is sometimes overlooked. You can look at this updated data to see how the data base has changed as a result of the simulation.

## *2.10 Preparing Tables and Graphs for a Report*

When you have run some simulations and analysed the simulation results, the next step is usually to write a report containing a selection of the results in tables or graphs.

This section describes how you can transfer simulation results into a spreadsheet program, such as Microsoft Excel. You can use the editing capabilities of the spreadsheet program to produce beautiful tables and graphs. These tables and graphs can be copied and pasted into your project report.

All the examples below start with the Solution file for Stylized Johansen **SJLB.SL4** created in the example simulation in section 2.4.6 or 2.4.7 for WinGEM or in section 2.6.2 or 2.6.3 for Unix/Command prompt.

If you are working on a PC, Example 1 illustrates how to copy directly from the Windows program ViewSOL into the spreadsheet program.

Examples 2 and 3 use the GEMPACK program SLTOHT which is available for both PC and Unix/Command prompt. SLTOHT is documented in chapters 8 and 9 of GPD-4.

SLTOHT produces a CSV (Comma Separated Value) file. This is a text file which can be opened in your spreadsheet program to import the numbers generated by the simulation. Even big arrays with many rows and columns can be moved to your report. There is no need to retype any of the numbers in an array. However you may wish to reformat them in some way (for example, change the number of decimal places).

Once you have suitable tables in your spreadsheet program, you can use the graphing capabilities of the program to create graphs of the results.

**Example 1 Copying from ViewSOL to Spreadsheet and WordProcessor (Windows PC only)**

In Windows, open file SJLB.SL4 in ViewSOL. Select ***Format*** and then select **Arrange vectors by size and set**.

In Contents screen, click on the line

**Vectors size: 2 SECT 4**

The Data Window shows the results for all vector variables which range over the set SECT.

In the ViewSOL menu, select ***Export | Copy***... This copies the table of numbers to the clipboard.

Open your spreadsheet program and paste into a new spreadsheet. Use the spreadsheet editing to make the table ready for the report. Copy the table from your spreadsheet and paste into it your word processor, in the usual Windows way, into the report document.

You should see a table like the following.

| | p_DVHOUS | p_PC | p_XCOM | p_XH |
|---|---|---|---|---|
| s1 | 5.8853 | 0 | 5.8853 | 5.8853 |
| s2 | 5.8853 | -0.9486 | 6.8993 | 6.8993 |

ViewSOL has many different Formats that you can use to set up the data to export. Consult the ViewSOL On-line help for details.

**Example 2 Using the GEMPACK Program SLTOHT and Option SSS**

In your text editor, create the file **sj1.map** which contains just the two lines:

```
p_xcom
p_xf
```

This is an example of a Spreadsheet Mapping file (see sections 8.3 and 9.1 of GPD-4) used by the program SLTOHT to pick out particular variables on the Solution file.

**If you are working in WinGEM**, select from the main WinGEM menu

***Other tasks... | Solution file to Header/Text (SLTOHT)***

Click on the ***Select*** button and choose the Solution file SJLB.SL4.

Choose to print **Totals solutions**. Click the ***Ok*** button.

In the SLTOHT window, select from the menu,

***Options | SLTOHT Options***

A screen of SLTOHT option choices will appear. Click on

**SSS Short SpreadSheet output**

and select a Comma as separator. (A comma is the default choice.)

Click on ***Ok*** to accept these options and return to the main SLTOHT screen.

In the SLTOHT window, select from the menu,

***Options | Use mapping file***

and select the file SJ1.MAP.

***Run*** the program SLTOHT. This will create the CSV text file called SJLB.CSV. When the program has completed, ***View*** the text file SJLB.CSV, which is shown below (after the Unix/Command prompt case).

**If you are working at the Unix/Command prompt**, start SLTOHT running by typing

```
sltoht
```

and enter the responses:

**Input to SLTOHT for Short Spreadsheet Output**

```
SSS       ! Option SSE Spreadsheet with element labels
,         ! Data separator is a comma
-SHL      ! Do not include levels results
<carriage-return>    ! Finish option selection
sjlb      ! Name of Solution file
c         ! Cumulative totals
y         ! Yes, use an existing Spreadsheet mapping file
sj1.map   ! Spreadsheet mapping file name
sjlb.csv   ! Name of Output file (CSV)
```

Look at the file **sjlb.csv** in your text editor.

The output in file **sjlb.csv** should look like the box below with the labels and numbers separated by commas. Only the values for variables p_XCOM and p_XF are shown because these were the only two variables in the Spreadsheet Mapping file.

**CSV file SJLB.CSV**

```
Solution,sjlb,
 p_XCOM(s1),  5.8852696    ,
 p_XCOM(s2),  6.8992925    ,
 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,
```

Start your spreadsheet program (for example Excel) and open the file SJLB.CSV (as a text file with commas for separators).[33] If you format the number cells to show three decimal places, you get a neat spreadsheet table with the labels in one column and the values in the second column. (In a report you would probably want to replace the column of labels with more meaningful labels.)

**Document Table**

| Solution | sjlb |
|---|---|
| p_XCOM(s1) | 5.885 |
| p_XCOM(s2) | 6.899 |
| p_XF(labor:s1) | 10.000 |
| p_XF(capital:s1) | 0.000 |
| p_XF(labor:s2) | 10.000 |
| p_XF(capital:s2) | 0.000 |

### Example 3 Using the Program SLTOHT and Option SSE

There are various different options available in SLTOHT used to produce different kinds of tables as described in chapter 9 of GPD-4. SSE stands for Spread Sheet with Element labels. This refers to a table of results using the element names as row and column labels.

In your text editor, create the file **sj2.map** which contains just the two lines:

```
p_xcom : p_pc
p_xc
```

**If you are using WinGEM**, in the SLTOHT window, select from the WinGEM menu,

***Options | SLTOHT Options***

A screen of SLTOHT option choices will appear. Click on

**SSE Spreadsheet output with element labels**

and select a Comma as separator. (A comma is the default choice.)

Click on ***Ok*** to accept these options and return to the main SLTOHT screen.

In the SLTOHT window, select from the menu,

***Options | Use mapping file***

[33] If you are working on a Unix machine and your spreadsheet program in on a Windows PC, you will need to move the (text) file sjlb.csv to your PC, for example, via FTP.

and select the file SJ2.MAP.

***Run*** the program SLTOHT.

SLTOHT will tell you that, by default, this will produce output file SJLB.CSV which already exists. [You created it in Example 2 above.] Click on **Yes** to say that you wish to change the name of the output file, and choose the name **SJLB2.CSV**.

When the program has completed, ***View*** the text file SJLB2.CSV, which is shown below (after the Unix/Command prompt case).

**If you are working at the Unix/Command prompt**, start SLTOHT running by typing

```
sltoht
```

and enter the responses:

**Input to SLTOHT for Spreadsheet Output with Element Names**

```
SSE       ! Option SSE Spreadsheet with element labels
,         ! Data separator is a comma
-SHL      ! Do not include levels results
<carriage-return>    ! Finish option selection
sjlb      ! Name of Solution file
c         ! Cumulative totals
y         ! Yes, use an existing Spreadsheet mapping file
sj2.map   ! Spreadsheet mapping filename
sjlb2.csv  ! Name of Output file (CSV)
```

Open the file **sjlb2.csv** in your spreadsheet program. The values for variables p_XCOM and p_PC are shown side by side and then the array p_XC is shown below, all with element labels.

| | | | | | |
|---|---|---|---|---|---|
| ! Table of 2 variables. | | | | | |
| | | | | | |
| | p_XCOM | p_PC | | | |
| s1 | 5.885 | 0.000 | | | |
| s2 | 6.899 | -0.949 | | | |
| | | | | | |
| ! Variable p_XC # Intermediate inputs of commodity i to industry j # | | | | | |
| ! Variable p_XC(SECT:SECT) of size 2x2 | | | | | |
| | | | | | |
| | s1 | s2 | | | |
| s1 | 5.885 | 5.885 | | | |
| s2 | 6.899 | 6.899 | | | |

You can import either of these tables into your word processor.

### 2.10.1 Graphs

Once you have suitable tables in your spreadsheet program, you can use the graphing capabilities of the program to create graphs of the results.

The Charter program supplied with GEMPACK (see section 2.2.1 of GPD-4) can be used to produce simple graphs.

## *2.11 Changing the Closure and Shocks*

You can carry out several simulations on the same model by changing the closure and/or the shocks in the Command file.

Note that, if you change the closure and/or shocks, but do not change the model (that is, do not change the TABLO Input file), you do not need to repeat Step 1 (running TABLO) in section 2.5. You only need to do Steps 2 and 3 there.

The following example shows you how to make a new Command file in the text editor and then run another simulation using SJ.EXE (or alternatively GEMSIM).

The new simulation is to increase the price of labor by 3 per cent and to increase the supply of capital by 10 per cent. In order to increase the price of labor, the variable p_PF("labor") needs to be exogenous. You need to change the closure and also apply these different shocks.

To change the command file SJLB.CMF, copy it to a new name SJLB2.CMF as follows:

- **If you are working in WinGEM**, in the main WinGEM menu, choose ***File | Edit File...*** then open the file SJLB.CMF. Click on ***File | Save As...*** and save the file under the new name SJLB2.CMF.
- **If you are working at the Unix/Command prompt**, copy the file sjlb.cmf to sjlb2.cmf (via the command "cp sjlb.cmf sjlb2.cmf" on a Unix machine).

Then use the text editor to modify this file, following the steps below.

(1) In the original closure, both components of p_XFAC (supplies of labor and capital) are exogenous. Here you keep the supply of capital exogenous, but set the price (rather than the supply) of labor exogenous. [p_PF is the variable in the model denoting the percentage change in the price of the factors, labor and capital.]

Find the statement

**exogenous p_xfac ;**

and change this to

**exogenous p_pf("labor") p_xfac("capital") ;**

(Be careful not to leave a space between the variable name p_pf and the bracket. It does not matter if you use upper or lower case, or a mixture, in Command files.)

(2) Shock p_pf("labor"), the price of labor, by 3 per cent and shock p_xfac("capital"), the supply of capital, by 10 per cent.

You will need **two** separate shock commands:

**shock p_pf("labor") = 3 ;**

**shock p_xfac("capital") = 10 ;**

[Remember to put a semicolon **;** after each statement.]

(3) Change the verbal description to describe the new closure and shocks. [This starts after "verbal description =" and ends with a semi-colon ";". There can be several lines of text in it.]

Exit from the editor after saving your changes.

**If you are working in WinGEM:**

If you have a Source Code version, click on ***Simulation | Run TG program...*** and then ***Select*** the TG Executable to be SJ.EXE.

If you have an Executable Image version, click on ***Simulation | GEMSIM Solve...*** .

Now both cases continue in the same way.

***Select*** the Command file SJLB2.CMF.

***Run*** the program with this Command file SJLB2.CMF. This is Step 2 of the simulation steps listed in section 2.5.

If there are errors when this runs, you will see a window headed "Error during Simulation". To correct the errors in the Command file, click on the button ***Edit Command file*** to use split screen editing. The Command file SJLB2.CMF will be shown in the top part of the screen and the LOG file in the bottom part of the screen. The errors will be marked in the LOG file, usually near the end of the file. When you have identified what is causing an error, you must make the appropriate change in the Command file in the top part of the screen (not the LOG file). The ***Next error*** button may help you to find errors. [If you have problems identifying or correcting the errors, you can find more assistance in section 3.9.2 below.] When you have corrected all errors, use ***File | Exit*** and save the changes you have made to the Command file SJLB2.CMF. Then close the error window by clicking on ***Close*** in it. Now click on ***Run*** to run the simulation again.

When SJ.EXE (or GEMSIM) has run successfully, click on ***Go to GEMPIE*** and then run GEMPIE to prepare a Print file of the solution (Step 3 of the simulation). Look at the results of the simulation in the Print file SJLB2.PI5. [Alternatively, click on ***Go to ViewSOL*** and use ViewSOL to look at the results.]

**If you are working at the Unix/Command prompt**,

Edit the Command file **sjlb2.cmf** to make the changes above.

To run the simulation, type in at the Command prompt:

```
sj  -cmf  sjlb2.cmf
```
or

```
gemsim -cmf sjlb2.cmf
```

View the Log file to see if there are any errors.

If there are errors, the errors will be marked in the LOG file (which is probably called **sjlb2.log**), usually near the end of the file. When you have identified what is causing an error, you must make the appropriate change in the Command file (not the LOG file). After you correct an error, rerun the simulation. [If you have problems identifying or correcting the errors, you can find more assistance in section 3.9.2 below.]

When sj (or gemsim) has run successfully, run **gempie** with the Solution file **sjlb2.sl4** to prepare a Print file **sjlb2.pi5** as in Step 3 for simulations (see section 2.6).

## *2.12 How to Carry out Several Simulations At Once*

The GEMPACK program SAGEM can be used to carry out a simulation, and indeed, as we shall see in this section, several simulations at once. However these simulations produce somewhat less accurate solutions than the multi-step calculations carried out earlier, since the linearized equations of the model are only solved once.

Nonetheless, carrying out these simulations, which are referred to as **Johansen simulations**[34], can be quite revealing.

In many cases, the results are sufficiently accurate to produce the right qualitative results. Being able to compute several such solutions as quickly as one multi-step solution has its advantages, especially for a new model whose behaviour you are just beginning to understand.

The starting point is always the Equations file for the model which is produced by running the TABLO-generated program SJ.EXE or GEMSIM. The **Equations file** contains the numerical linearized equations of the model. (This is explained in more detail in section 2.13.1 below.)

### 2.12.1 Preparing an Equations File for Use by SAGEM

An Equations file for Stylized Johansen can be created by running the TABLO-generated program SJ.EXE or GEMSIM and taking inputs from the Command file **SJEQ.CMF** shown below.

**Figure 2.12.1: The GEMPACK Command File SJEQ.CMF**

```
! The following GEMPACK Command file (usually called SJEQ.CMF)
!  produces an Equations file for Stylized Johansen.
! It also creates an Environment file for the standard closure.

!  Auxiliary files for model
auxiliary files = sj ;

! Data files
file iodata = sj.dat ;

! Equations file information
equations file = sj ;  ! creates a new Equations file
  model = sj ;
  version = 1 ;
  identifier = Stylized Johansen. Standard data. ;

! Don't carry out a simulation
simulation = no ;

! End of Command file
```

The statements

```
equations file = sj ;
  model = sj ;
  version = 1 ;
  identifier = Stylized Johansen. Standard data. ;
```

specify the name of the Equations file SJ.EQ4 (the suffix .EQ4 is added automatically by the TABLO-

[34] The name pays tribute to Johansen who pioneered this way of obtaining useful, approximate solutions of general equilibrium models around 1960.

generated program SJ.EXE or GEMSIM), the model name, the version number and a model identifier (which can be up to 60 characters long).

The statement

```
simulation = no ;
```

indicates that a simulation is not being carried out (so the software expects no "shock" statements).

Whether you are using WinGEM or the Command prompt method for running GEMPACK, proceed as in Step 2 of the example simulation above but use the Command file SJEQ.CMF instead of SJLB.CMF. This will create the new Equations file SJ.EQ4.[35]

### 2.12.2 Running Several Johansen Simulations using SAGEM

You will use the Command file **SJLBJ.CMF** for running SAGEM to carry out these simulations. This gives shocks of 1 percent to supplies of both labor and capital. We suggest that you first carry out the simulation, as described below. We will then discuss the results and tell you about the statements in this Command file.

If you are using WinGEM, proceed as in section 2.12.2.1. If you are working at the Unix/Command prompt, proceed as in section 2.12.2.2.

#### *2.12.2.1 WinGEM: Running Several Johansen Simulations using SAGEM*

Click on ***Simulation | SAGEM Johansen Solve...*** in the main WinGEM window.

***Select*** the file SJLBJ.CMF and ***Edit*** to check that this Command file tells SAGEM

(i) to use the Equations file SJ.EQ4 (made by the TABLO-generated program SJ.EXE or GEMSIM in previous example in section 2.12.1 above)

```
use equations file sj ;
```

(ii) to give uniform shocks of 1 percent to supplies of the two factors using the command:

```
shock p_xfac = uniform 1 ;
```

(iii) to retain all individual column results using the command:

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

(iv) Note that, although the Solution file is the most important output from the simulation, the name of the Solution file is not included in the Command file. This is because of the convention that the Solution file is given the same name as that of the Command file unless the Command file specifies otherwise. [Here, since the Command file is called SJLBJ.CMF, the Solution file will be called SJLBJ.SL4 – the same name "SJLBJ" but with the standard Solution file suffix ".SL4".]

Then ***File | Exit*** to exit from the Command file.

To run this Johansen simulation, ***Run*** the program SAGEM.

---

[35] An alternative way of creating the Equations file would be to add the lines

```
equations file = sj ;
  model = sj ;
  version = 1 ;
  identifier = Stylized Johansen. Standard data. ;
```

to the file SJLB.CMF and rerun the simulation itself.

Click on ***Go to GEMPIE*** (not *Go to ViewSOL* this time since ViewSOL cannot show individual column results – see section 10.2.5 of GPD-3) to create a print file of the results. You will be asked whether you want the individual column solutions or the totals solution. Select the ***individual column*** solutions.

***Run*** GEMPIE to create the Print file SJLBJ.PI5. ***View file*** to see the results of the simulation. Check that your results are the same as those shown in Table 2.12.3 below.

How many columns of results are there? Each column contains a different solution. What are the shocks associated with each of the columns? [The first column contains the results of a 1% increase in the supply of labor, holding the supply of capital fixed. We discuss these results in more detail in section 2.12.3 below.]

This completes the Johansen simulation. Use ***File | Exit*** to close the GEMPIE and SAGEM windows.

Now skip ahead to section 2.12.3 where we discuss the results.

#### *2.12.2.2 Unix/Command Prompt: Running Program SAGEM*

When running SAGEM you specify the Equations file to use, the closure and shocks.

The following run of SAGEM carries out several Johansen simulations with the Stylized Johansen model.

Start SAGEM running by entering the command

```
sagem
```

(or whatever variation is required on your system). When SAGEM prompts you, give the two responses below.

```
cmf           ! Take inputs from a GEMPACK Command file
sjlbj.cmf     ! The name of the Command file
```

This should create the Solution file usually called **sjlbj.sl4**. (The last "j" in the name is a reminder that this contains Johansen solutions.) To see the results, you should run GEMPIE, for example by typing

```
gempie
```

(or otherwise, as required on your system), and then giving the responses below.

```
<carriage-return>       ! Take the default program options
sjlbj                   ! Solution file (from SAGEM run)
i                       ! Print individual results from Solution file
a                       ! Print all available columns
a                       ! Print all available rows
<carriage-return>       ! Default name (SJLBJ) for Print file
All shocks 1            ! Heading for Print file
5                       ! Decimal places
```

**User Input to GEMPIE**

This should create the GEMPIE Print file **sjlbj.pi5**, which you should print or look at in an editor.

Check that your results are the same as those shown in Table 2.12.3 below.

How many columns of results are there? Each column contains a different solution. What are the shocks associated with each of the columns? [The first column contains the results of a 1% increase in the supply of labor, holding the supply of capital fixed. We discuss these results in more detail in section 2.12.3 below.]

This completes the Johansen simulation.

### 2.12.3 The Results of these Johansen Simulations

Some of the results from GEMPIE Print file SJLBJ.PI5 are shown in Table 2.12.3.

Page 1 of the results (these are on the second last page of the Print file) should be (approximately) as shown in Table 2.12.3. (Page 2 of the results gives the results for variables p_DVFACIN and p_DVHOUS, which we do not show in Table 2.12.3.)

The results shown in Table 2.12.3 are **individual column results** because each column shows the approximate effect of the exogenous movement in one of the exogenous variables shocked in the simulation. The first column shows the effects on the endogenous variables of a 1 per cent increase in the supply of labor (with no change in the supply of capital) while the second column shows that of a 1 per cent increase in just the supply of capital. (The third column is the total of these two results.)

Because these are results of a Johansen simulation, the results are not as accurate as the single result produced in running the TABLO-generated program or GEMSIM above. However, the advantage of Johansen results is that they can be scaled and combined. For example, the Johansen results of a 10 per cent increase in the labor supply can be inferred by multiplying the results of a 1 per cent increase (column 1 in the Print file) by 10. You should do this and compare the results with those of the multi-step simulation in section 2.7 above. (For example, the extrapolated result for household expenditure 'p_Y' is 5.8853 while from the Johansen simulation the corresponding, less accurate, result is 6.0.)

You can also combine columns to estimate the cumulative effect of any shocks to supplies of labor and capital. For example, to estimate the cumulative effect of an increase of 5 per cent in the supply of labor and a decrease of 10 per cent in the supply of capital, simply multiply the results in the two columns above by 5 and -10 respectively and add the results.

This is the main reason for using SAGEM rather than running SJ.EXE or GEMSIM. SAGEM can produce the Johansen results of several simulations[36] more quickly (that is, using less computing resources) than one single, more accurate, multi-step simulation result.

[36] The run of SAGEM in question can be said to produce the results of any pair of increases/decreases in the supply of labor and capital since the Johansen results can be multiplied and combined. (Note that multi-step simulation results cannot be combined in this way.)

**Table 2.12.3 : Individual Column Results From GEMPIE (SJLBJ.PI5)**

```
PAGE 1      All shocks 1
                              p_XFAC          p_XFAC          TOTALS
                                1               2
                             1.00000         1.00000


p_Y   Total nominal household expenditure
    1                        0.60000         0.40000         1.00000


p_PC (SECT)  Price of commodity i
    1 s1                     0.00000*        0.00000*        0.00000*
    2 s2                    -0.10000         0.10000         0.00000


p_PF (FAC)  Price of factor f
    1 labor                 -0.40000         0.40000         0.00000
    2 capital                0.60000        -0.60000         0.00000*


p_XCOM (SECT)  Total demand for (or supply of) commodity i
    1 s1                     0.60000         0.40000         1.00000
    2 s2                     0.70000         0.30000         1.00000


p_XH (SECT)  Household demand for commodity i
    1 s1                     0.60000         0.40000         1.00000
    2 s2                     0.70000         0.30000         1.00000


p_XC (SECT,SECT)  Intermediate inputs of commodity i to industry j
   (-,s1) results where '-' is in set 'SECT'.
    [1] 1 (s1,s1)            0.60000         0.40000         1.00000
    [2] 2 (s2,s1)            0.70000         0.30000         1.00000
   (-,s2) results where '-' is in set 'SECT'.
    [3] 1 (s1,s2)            0.60000         0.40000         1.00000
    [4] 2 (s2,s2)            0.70000         0.30000         1.00000


p_XF (FAC,SECT)  Factor inputs to industry j
   (-,s1) results where '-' is in set 'FAC'.
    [1] 1 (labor,s1)         1.00000         0.00000*        1.00000
    [2] 2 (capital,s1)       0.00000*        1.00000         1.00000
   (-,s2) results where '-' is in set 'FAC'.
    [3] 1 (labor,s2)         1.00000         0.00000*        1.00000
    [4] 2 (capital,s2)       0.00000*        1.00000         1.00000


p_DVCOMIN (SECT,SECT)  Dollar value of inputs of commodity i to industry j
   (-,s1) results where '-' is in set 'SECT'.
    [1] 1 (s1,s1)            0.60000         0.40000         1.00000
                             Next 1 component(s) are the same as 1.
   (-,s2) results where '-' is in set 'SECT'.
    [3] 1 (s1,s2)            0.60000         0.40000         1.00000
                             Next 1 component(s) are the same as 3.
```

### 2.12.4 The Command File SJLBJ.CMF for these Johansen Simulations

The GEMPACK Command file for SJLBJ.CMF used to run SAGEM above is shown in full in Figure 2.12.4 below. Notice that this uses the Equations file SJ.EQ4 produced during the run of SJ.EXE or GEMSIM in section 2.12.1 above.

**Figure 2.12.4: The GEMPACK Command File SJLBJ.CMF for SAGEM**

```
!
! The following GEMPACK Command file (usually called SJLBJ.CMF)
!  will run SAGEM to carry out two Johansen simulations with
!  the Stylized Johansen model.

! Simulation part
use equation file sj ;  ! Uses existing Equations file

solution file = sjlbj ;

! Closure
exogenous p_XFAC ;
rest endogenous ;

! Shock
shock p_xfac = uniform 1 ;

verbal description =
Stylized Johansen model. Standard data and closure.
1 per cent increases in supplies of labor and capital.
Johansen solutions. ;

! Retain all individual column results
individually-retained exogenous %all ;

! End of Command file
```

The statement

**`individually-retained exogenous ;`**

tells SAGEM that we want all the individual column results on the Solution file. The other statements in the Command file should be self-explanatory. (Full documentation of the statements allowed in GEMPACK Command files for running SAGEM can be found in section 18.2 in GPD-3.)

When you run SAGEM, the Solution file produced can contain several individual column results and/or cumulative results[37] and subtotals. More information about these is given in chapter 10 of GPD-3.

Another important difference between running SAGEM and running SJ.EXE or GEMSIM to carry out a simulation relates to the updated data. SAGEM does not, and cannot, produce updated data (which is why there is no "updated file iodata = … ;" statement in the Command file above), whereas TABLO-generated programs and GEMSIM do produce updated data (see section 2.9 above).

[37] To see the cumulative result (the combined effect of increases of 1 per cent in the supplies of both factors), re-run GEMPIE but this time give 't' for totals as your third response, and then continue the dialogue in the obvious way.

### 2.12.5 SAGEM - Advantages and Disadvantages

In summary, the advantages of using SAGEM are that:

- Several approximate simulations can be run very quickly.
- Results are usually qualitatively correct.
- You can combine results from different columns to produce a solution which also is a Johansen solution for the model.

The disadvantages of a one-step solution produced by SAGEM are that:

- Each result is only approximate.
- No updated data is calculated.
- No levels results are produced.

## *2.13 How Johansen and Multi-step Solutions are Calculated*

Johansen solutions are approximate results of a simulation. In contrast, multi-step solutions can be made arbitrarily accurate by taking enough steps. In this section we describe the main ideas involved in calculating these different solutions.

### 2.13.1 The Linearized Equations of a Model

Johansen solutions are calculated by solving the linearized equations[38] of the model once while multi-step solutions are obtained by solving these equations several times.

The system of linearized equations of any model can be written in the form

$$\mathbf{C\ z = 0} \qquad (1)$$

where

**C** is the n x m matrix of coefficients of the equations, known as the Equations Matrix (which is closely related to the Equations file – see section 2.12.1), [39]

**z** is the m x 1 vector of all the variables (usually in percentage change form) of the model,

**n** is the total number of equations, and
**m** is the total number of variables.

We call C the **Equations Matrix** of the model.[40] It is often useful to think of this matrix as a rectangular array or tableau[41] with the vector variables across the top and the equation blocks 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.

To illustrate this, part of the tableau for the 27 x 29 Equations Matrix C for the Stylized Johansen model (from the TABLO Input file SJ.TAB) is shown in Table 2.13.1 below.

Notice that we use the words "variable" and "equation" in two different senses. For example, we usually say that Stylized Johansen is a model with 29 variables and 27 equations, where we count as variables all the components of the vector variables and we count as equations all the individual equations in the equation blocks. In this sense, the number of variables is the number of columns in the Equations Matrix while the number of equations is the number of rows. Alternatively we may say that the TABLO Input file for Stylized Johansen has 11 variables (meaning vector variables) and 10 equations (meaning equation blocks). Usually the context will make clear which of these two meanings is intended.

---

[38] If some or all of the equations in the TABLO Input file are levels equations, TABLO automatically converts them to the associated linearized equations. (It does the differentiation symbolically.) The linearized equations are the ones solved by GEMSIM or the TABLO-generated program.

[39] The Equations file for the model is essentially this Equations Matrix C. The numerical values in C come from evaluating the symbolic linearized equations of the model (as held on the TABLO Input file, or as linearized by TABLO) by inserting the values from the initial data attached when the model is solved (as in Step 2 in section 2.5.1 or 2.5.2 above). This matrix C is often denoted by A(V) in DPPW.

[40] Some of the entries of the Equations Matrix are calculated for a version of the Stylized Johansen model in section 3.4.5 below.

[41] This explains the origin of the name TABLO.

**Table 2.13.1: Tableau of the Equations Matrix for Stylized Johansen**

| | | 1 | 2 | 2 | 2 | 4 | 2 |
|---|---|---|---|---|---|---|---|
| | | p_Y | p_PC | p_PF | p_XCOM.... | p_DVFACIN | p_DVHOUS |
| cols --> | | 1 | 2 3 | 4 5 | 6 7 | 24..27 | 28 29 |
| rows | | | | | | | |
| | 1 | | | | | | |
| Comin | 2 | | | | | | |
| 4 | 3 | | | | | | |
| | 4 | | | | | | |
| | 5 | | | | | | |
| Facin | 6 | | | | | | |
| 4 | 7 | | | | | | |
| | 8 | | | | | | |
| House | 9 | | | | | | |
| 2 | 10 | | | | | | |
| | | | | | | | |
| : | | | | | | | |
| : | | | | | | | |
| | | | | | | | |
| Numeraire | 27 | | | | | | |
| 1 | | | | | | | |

In general, n is less than m in the system of equations in (1) above, so when you carry out a simulation (Johansen or multi-step) you must specify

- (m-n) of the variables as exogenous and the remaining variables as endogenous, and
- shocks (usually percentage changes) to some of the exogenous variables.

For Stylized Johansen, the total number of variables (m) is 29 and the total number of equations (n) is 27, so we need 2 exogenous variables. We can shock either 1 or 2 of these exogenous variables.

The numerical values of some of the entries in the Equations Matrix can be seen in section 3.4.5 below.

### 2.13.2 Johansen Solutions

Johansen solutions are defined to be solutions obtained by solving the linearized equations of the model **just once**. Because the levels equations of the model are usually nonlinear, the results of this calculation are only approximations (sometimes good approximations and sometimes not-so-good) to the corresponding solution of the levels equations of the model.

Once the exogenous/endogenous split has been chosen, the system of equations **C z = 0** in (1) above, becomes .

$$\mathbf{A}\ z_1 = -\mathbf{D}\ z_2 \qquad (2)$$

where $z_1$ and $z_2$ are respectively the (column) vectors of endogenous and exogenous variables, A is **n x n** and D is **n x (m-n)**.

The matrix A is referred to as the **LHS Matrix** (Left Hand Side Matrix) of the simulation. The LHS matrix A consists of the columns of the Equations matrix corresponding to the endogenous variables in the given closure. Similarly the columns of the matrix D are just the columns of C corresponding to the exogenous variables in the closure. The shocks are the values to use for $z_2$. Once these are known, we have a system

$$\mathbf{A}\ z_1 = \mathbf{b} \qquad (3)$$

to solve (where the RHS vector **b** is an n x 1 vector equal to $-Dz_2$). It is the solution $\mathbf{z_1}$ of this matrix equation (3) which is the **Johansen solution** of the simulation.[42]

### 2.13.3 Multi-step Simulations and Accurate Solutions of Nonlinear Equations

The idea of a multi-step simulation is to break each of the shocks up into several smaller pieces. In each step, the linearized equations are solved for these smaller shocks. After each step the data, shares and elasticities are recalculated to take into account the changes from the previous step. In general, the more steps the shocks are broken into, the more accurate will be the results.

Figure 2.13.3 below makes this easy to visualise. In that figure we consider just one exogenous variable X (shown on the horizontal axis) and one endogenous variable Y (vertical axis); these are constrained to stay on the curve g(X,Y) = 0. We suppose that they start from initial values $X_0$,$Y_0$ at the point A and that X is shocked from value $X_0$ to value $X_1$. Ideally we should follow the curve g(X,Y)=0 in solving this. In a Johansen (that is, a 1-step) solution we follow the straight line which is a tangent to the curve at point A to reach point $B_J$ and so get solution $Y_J$.

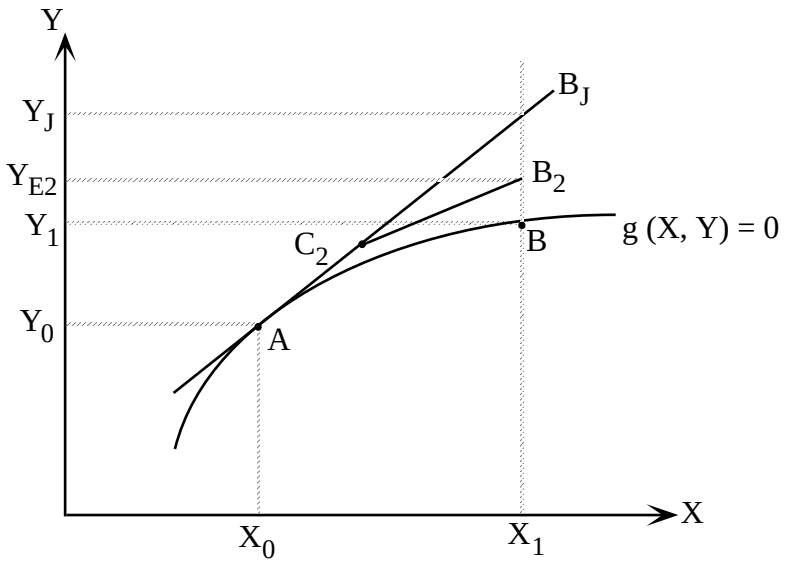



**Figure 2.13.3 Illustration of Euler's Method**

In **Euler's method** the direction to move at each step is essentially that of the tangent to the curve at the appropriate point. In a 2-step Euler solution (see Figure 2.13.3), we first go half way along this tangent to point $C_2$, then recompute the direction in which to move, and eventually reach point $B_2$, giving solution $Y_{E2}$. The exact solution is at B where Y has value $Y_1$. In a 4-step Euler simulation we follow a path of 4 straight-line segments, and so on for more steps.

The default method used by GEMPACK is **Gragg's method** which uses an even more accurate method than Euler's method for calculating the direction in which to move at each step. When the shocks are broken into N parts, Euler's method does N separate calculations while Gragg's method does N+1.

---

[42] The matrix A is usually sparse in the sense that most of its entries are zero. GEMPACK uses the Harwell Laboratory's sparse matrix routines (see section 12.1 of GPD-3) to solve (3). These solve (3) by calculating an LU decomposition of A (which is always more efficient than calculating the inverse of A). It is the sparsity of A which enables GEMPACK to handle such large models. See section 12.1 of GPD-3 for more details.

Usually the computational cost of this extra calculation is more than repaid by the extra accuracy obtained. (More information about Gragg's method can be found in section 12.2 of GPD-3. This also introduces the **midpoint method** which is similar to Gragg's method.)

So one way of increasing accuracy is to increase the number of steps. It turns out, however, that the best way to obtain an accurate solution is to carry out 2 or 3 different multi-step calculations with different numbers of steps and to then calculate the solution as an appropriate weighted average of these. This is what is meant by the **extrapolated solution**.

To illustrate this, we have shown below the different results for the percentage change in household expenditure 'p_Y' in the Stylized Johansen model for the simulation in section 2.1 above, in which labor supply is increased by 10 per cent and capital remains in fixed supply. The table below shows Euler and Gragg results for different step numbers and extrapolations based on them. Note that the exact result is 5.88528.

**Multi-step results for different methods and step numbers**

| Method | Number of steps | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 6 | 100 |
| Euler | 6.00000 | 5.94286 | 5.91412 | 5.90452 | 5.88644 |
| Gragg [43] | | 5.88675 | 5.88545 | 5.88529 | |

**Extrapolated results**

| | |
|---|---|
| From Euler 1,2-step results | 5.88571 |
| From Euler 1,2,4-step results | 5.88527 |
| From Gragg 2,4,6-step results | 5.88529 |

Note that, in this case,

- the 4-step Gragg result is more accurate than the 100-step Euler result, and
- the result extrapolated from 1,2,4-step Euler results is much more accurate than the 100-step Euler result (even though the latter takes about 100/7 times as long to compute).

These results are typical of what happens in general.

The general messages are:

1. Gragg's method is usually much more accurate than Euler's (for the same number of steps).
2. If in doubt, extrapolate.
3. Extrapolating from 3 different solutions is better than from 2.
   (For example, extrapolating from Gragg 2,4 and 6-step solutions is usually better than from just 4 and 6-step solutions.)

When you extrapolate, if you ask for an **Extrapolation Accuracy file** (usually an .XAC file),[44] this file shows how accurate the solution is for each endogenous variable. The separate columns show the results for the different multi-step solutions calculated, and the last column of results is the extrapolated result. When you extrapolate from 3 different multi-step results (which is what we recommend), the last two columns give conservative information about the accuracy of each result. (If they show M figures agreement, this means that the 2 different extrapolations based respectively on just the first two and just the first and third agree to this number of figures.)

---

[43] A 1-step Gragg calculation doesn't make much sense, so we have not shown a result for it.

[44] You can get an Extrapolation Accuracy file by including the statement "extrapolation accuracy file = yes ;" in your Command file – see section 2.8.1.

For example, for the 1,2,4-step Euler results for household expenditure 'p_Y' reported above for the SJLB.CMF simulation (see sections 2.4 and 2.6), the relevant line in the Extrapolation Accuracy file would be[45]

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

The results are the 1,2,4-step results and the extrapolation based on them. The comment "CX 4" is an abbreviation meaning that you can have confidence in the extrapolated result (this is the 'CX') and that the two extrapolations (the first based just on the 1,2-step results and the second based on the 2,4-step results) agree to 4 figures (or more). Note that the agreements are reported as figures, not decimal places. (For example 123.4567 and 123.4014 agree to 4 figures, but only one decimal place.) The abbreviations (such as 'CX') used on this file are explained at the top of the file. (The first "1" in the line displayed above means that this line refers to the first - in this case, the only - component of variable p_Y.) The "L5" at the end of this line means that you can be confident of 5 figures accuracy in the level of income Y. [See section 7.2.1 of GPD-3 for more details.]

At the end of the file is a summary (we refer to it as the **Extrapolation Accuracy Summary**) which states how many components fall into each category (EMA, CX etc). For a simulation with Stylized Johansen, this may look something like that shown below.

```
      SUMMARY OF CONVERGENCE RESULTS              Number  Min Figs Agree
      ------------------------------              ------  --------------
EMA   Last two results equal to machine accuracy     3         6
FC0   Fair confidence that the result is zero        2
CX    Confidence in the extrapolated result         22         2

  2 results are judged accurate to 2 figures.
  4 results are judged accurate to 3 figures.
  16 results are judged accurate to 4 figures.
  3 results are judged accurate to 6 figures.

 Above is for linearised variables.
 Below are for levels values of percent-change and change results.

  1 results are judged accurate to 4 figures.
  21 results are judged accurate to 5 figures.
  5 results are judged accurate to 6 figures.

   (The summary above covers the XAC-retained variables.)
```

The first part is a summary of the number of times different comments (in the example above, "EMA", "FC0" and "CX") have been used for the different results. The second part tells how many results (linearised, then levels) have been judged accurate to different numbers of figures.

More details about Extrapolation Accuracy Summaries and Extrapolation Accuracy files are given in section 7.2 of GPD-3.

The only restriction on step numbers is that, for Gragg's method and the midpoint method, the step numbers must either be all odd or all even (for example, 2,4,6 or 3,5,7). Note also that a 1-step Gragg or midpoint is a little unusual and is probably best avoided (since it is more like Euler than Gragg or midpoint).

More details are given in section 12.2 of GPD-3 and some of the theory behind multi-step methods can be found in Pearson (1991).

---

[45] You can check this by looking at the Extrapolation Accuracy file SJLB.XAC produced when you ran the simulation.

## *2.14 GEMPACK Programs – An Overview*

As you have already seen, GEMPACK is not a single program. There are a number of different programs making up GEMPACK.

In this section we give a brief overview of these different programs, indicating what you use the programs for.[46] We group these by task.

You have not yet used all of the programs below. As you become more experienced with GEMPACK, you may need to come back to this section to check out those programs you have not used.

### 2.14.1 Working with TABLO Input Files

**TABmate**. Windows program TABmate is a special editor for TABLO Input files. Use TABmate to create and modify the theory of a model – that is, the TABLO Input file for the model.

**TABLO**. Run TABLO to convert the TABLO Input file for a model into a TABLO-generated program or Auxiliary files for GEMSIM. You must do this before you can carry out a simulation with a model. [See Step 1 in section 2.5.]

In fact TABmate runs the program TABLO in the background. So TABmate and TABLO can perform very similar functions. You can use either TABLO or TABmate to carry out Step 1 for simulations (see section 2.5).

### 2.14.2 Carrying out Simulations

You can use either GEMSIM or the relevant TABLO-generated program to carry out Step 2 for simulations (see section 2.5). Which you use depends on how you ran TABLO in Step 1 (see section 2.5.1 or 2.5.2 above).

**GEMSIM**. Run GEMSIM to carry out a simulation. Specify the Auxiliary files, starting data files, closure and shocks in a Command file.

**TABLO-generated program**. Run the executable image of the TABLO-generated program to carry out a simulation. Specify the starting data files, closure and shocks in a Command file. You can only create TABLO-generated programs if you have a Source-code version of GEMPACK.

**SAGEM**. Use SAGEM to carry out Johansen simulations. First you must prepare an Equations file using GEMSIM or the TABLO-generated program. Specify the closure and shocks in a Command file. SAGEM cannot produce accurate solutions but it can produce several Johansen (approximate) solutions in one run. [See section 2.12.]

### 2.14.3 Looking at, and Processing, Simulation Results

**ViewSOL**. A Windows program for looking at results. [See Step 3 in section 2.5.]

**GEMPIE**. An alternative way of looking at results. [See Step 3 in section 2.5.]

**AnalyseGE**. Windows program AnalyseGE is used to analyse simulation results. In the process, AnalyseGE gives you access to simulation results and to the values of items in the data base. [See section 2.6 of GPD-4 for an introduction to AnalyseGE. A hands-on introduction to AnalyseGE in the context of a simulation with Stylized Johansen can be found in section 6.1 of GPD-8.]

---

[46] The programs are listed in section 1.7. All of these programs can be used on Windows PCs, but the Windows programs (see point 4 in section 1.7) are not available on Unix machines.

**SLTOHT**. SLTOHT can be used to produce tables of results which you can include in reports. SLTOHT can also be used to convert simulation results (which are on a Solution file) to a Header Array file. [See section 2.10 above and chapters 8 and 9 of GPD-4 for information about SLTOHT.]

When you report simulation results, you will probably use other standard tools (including spreadsheet programs such as Excel) for making tables and graphs.

### 2.14.4 Working with Data

In GEMPACK data for models are usually held on Header Array files. Because these are binary files, you need programs to work with the data in this form.

**ViewHAR**. A Windows program for looking at data on Header Array files. If you are working on a Windows PC, you used ViewHAR to look at original and updated data in sections 2.4.4 and 2.9 above. ViewHAR can also be used to create a Header Array file and to modify data on a Header Array file (see section 4.5 below and section 2.2 of GPD-4).

**SEEHAR**. An alternative way of looking at data on a Header Array file. If you are working at the Unix/Command prompt, you used SEEHAR to look at the original and updated data in sections 2.6.1 and 2.9 above. SEEHAR can also be used on Windows PCs (see, for example, section 2.4.17 of GPD-8). [SEEHAR is documented in section 4.1 of GPD-4.]

**MODHAR** can be used to create a Header Array file and to modify data on a Header Array file. [See sections 4.2 and 4.3 below and chapter 3 of GPD-4.]

**CMPHAR** can be used to compare the data on two Header Array files. [See section 4.2 of GPD-4.]

**CMBHAR** can be used to combine the data on two or more Header Array files (for example, data representing two or more years). [See section 4.4 of GPD-4.]

Data comes from various sources. You may use other standard tools and programs (including spreadsheet programs such as Excel) when working with data.

### 2.14.5 Environments for Modelling on Windows PCs

**WinGEM**. Windows program WinGEM provides an environment for carrying out modelling and associated tasks. If you are working on a Windows PC, you have been using WinGEM throughout this chapter.

**RunGEM.** Windows program RunGEM provides an environment for carrying out simulations with a fixed model. Users with little experience of modelling can carry out simulations by choosing just the closure and shocks. [See section 2.5 of GPD-4 and chapter 5 of GPD-8.]

### 2.14.6 Other Programs For Special Tasks

These are programs used for specialised tasks, usually by power users. You will not need to use any of them until you are more experienced with GEMPACK. These programs are documented in GPD-4.

**SUMEQ**. For looking at data on an Equations file. SUMEQ can also be used to diagnose homogeneity problems with a model. [SUMEQ is documented in chapter 13 of GPD-4.]

**SEENV**. For working with Environment files (which store the set of exogenous and endogenous variables in one closure for a model). [See chapter 12 of GPD-4.]

**RWHAR** and **MKHAR**. Programs which are used to convert Header Array files from one sort of computer (for example, a Windows PC) to another sort of computer (for example, a Unix machine). [See chapter 11 of GPD-4.]

**RunDynam** and variants including **RunMONASH** and **RunGDYN**. Windows interfaces for use with recursive dynamic models such as MONASH and dynamic GTAP. [See section 2.7 of GPD-4.]

**ACCUM** and **DEVIA**. These are used for working with recursive dynamic models such as MONASH which are solved annually over a period of several years. ACCUM and DEVIA collect the results for several years. [See chapter 10 of GPD-4.]

## *2.15 Different GEMPACK Files*

As you have seen, GEMPACK programs produce and use several different types of files (for example, Solution files and Header Array files). New users often ask us "Why are there so many different files?".

In this section we give more details about these different files and how they are used.

In our experience, some users are keen to have detailed information about this topic while others are not very interested. Since a detailed knowledge about this topic is not essential for doing effective modelling, **you should feel free to skip this section** or to skim it very quickly. You can always refer back to it later on, if and when you need to.

A table summarising the different files is given in section 2.15.7 below.

### 2.15.1 The Most Important Files

We begin with the most important files, namely TABLO Input files, data files, Command files and Solution files, all of which you have met earlier in this chapter.

**TABLO Input files.** These contain the theory (equations etc) for a model. Alternatively they may be for data manipulation. These files have suffix **.TAB** . These files are inputs to the program TABLO. The program TABmate (see section 2.4 of GPD-4) can be used to create and or modify these files.

**Data files.** These may be Header Array files or text files. The suffix is not prescribed by the software, though suffixes **.HAR** and **.DAT** are commonly used. Data files can be inputs to a simulation (the original input-output data, the parameters) and updated versions are output from a simulation. Updated data files are often given the suffix **.UPD** . Chapter 4 below contains an introduction to the different ways in which you can create data files and modify data on existing data files.

**Command files.** These contain the details of a simulation, including closure, shocks, starting data and solution method (see section 2.8). The suffix is not prescribed by the software, though **.CMF** is very commonly used.[47] Command files are inputs for GEMSIM, TABLO-generated programs and SAGEM. The statements allowed in Command files are documented in chapter 18 of GPD-3. The statements allowed in SAGEM Command files are slightly different from those allowed for GEMSIM and TABLO-generated programs (see section 18.2 of GPD-3).

**Solution files.** These are the main outputs from a simulation. They contain the change or percentage change results for all the linearized variables. They may also contain levels results. These files have suffix **.SL4** . They are inputs to the various post-simulation software programs including ViewSOL, GEMPIE, AnalyseGE and SLTOHT. Solution files are documented in chapter 8 of GPD-3.

**Equations files.** These contain numerical versions of the linearized equations of a model. They are usually only produced when you wish to use SAGEM to obtain several Johansen (approximate) solutions of a model, as explained in section 2.12.1 above. You may also create an Equations file if your model is not homogeneous in prices or quantities (see section 13.2 of GPD-4). Equations files have suffix **.EQ4** . Equations files are documented in chapter 9 of GPD-3.

---

[47] There are advantages in using this suffix .CMF (see section 2.5.4 of GPD-3).

**Shock files.** Sometimes it is necessary to shock the different components of a variable by different amounts. If the variable has a large number of components, or if the shocks are calculated by a program, it is convenient to put the numerical values of the shocks onto a so-called "shocks file". This file may be a text file or a Header Array file. The suffix for shocks files is not prescribed by the software, though often **.SHK** is used. The use of shock files is documented in sections 5.5 to 5.8 of GPD-3.

**Solution Coefficients (SLC) files.** These are output from a simulation. They contain the pre-simulation values of all data and of all Coefficients in the TABLO Input file for the model. These files have suffix **.SLC** and have the same name (apart from suffix) as the Solution file from the simulation.[48] The program AnalyseGE reads both the Solution and SLC file when you analyse the results of a simulation.[49] SLC files are documented in section 8.4 of GPD-3.

**Extrapolation Accuracy files.** You can ask for an Extrapolation Accuracy file to be produced when you extrapolate from 3 separate multi-step calculations (as you have seen in section 2.13.3 above).[50] These show estimates as to how accurate the different simulation results are. These files have suffix **.XAC** and have the same name (apart from suffix) as the Solution file from the simulation. Extrapolation Accuracy files are documented in section 7.2 of GPD-3.

### 2.15.2 Files for Communication Between Programs

There are a number of files which are used for communication between programs. The most important of these are the Auxiliary files which allow communication between TABLO and GEMSIM or the TABLO-generated program.

**Auxiliary files.** These are either

- **Auxiliary files for a TABLO-generated program.** These are produced when TABLO writes a TABLO-generated program (see section 2.5.1 above). These files have suffix **.AXS** (Auxiliary Statement file) and **.AXT** (Auxiliary Table file).
- **GEMSIM Auxiliary files.** These are produced when TABLO writes output for GEMSIM (see section 2.5.2 above). These files have suffix **.GSS** (GEMSIM Statement file) and **.GST** (GEMSIM Table file).

**TABLO-generated program (Fortran file).** This is the Fortran file called the TABLO-generated program which is produced by TABLO (see section 2.5.1 above). This file has suffix **.FOR** on Windows PCs and has suffix **.f** on Unix machines. Auxiliary files (AXS and AXT files) are always produced at the same time. In Step 1(b) for simulations (see section 2.5.1), this program is compiled and linked to produce the executable image of the TABLO-generated program.[51] It is this executable image which is run to carry out a simulation.

When the executable image of a TABLO-generated program is used to carry out a simulation, the Auxiliary files (AXS and AXT files) are required. They communicate vital information from the TABLO Input file to the simulation.

---

[48] If the Solution file is called SJLB.SL4 then the associated SLC file is called SJLB.SLC .

[49] Variants of SLC files are UDC, AVC and CVL files – see sections 8.5 and 8.6 of GPD-3 for details.

[50] To generate an Extrapolation Accuracy file, put the statement "extrapolation accuracy file = yes ;" into your Command file.

[51] The executable image has suffix .EXE on Windows PCs but no suffix on Unix machines. [On Unix, if the TABLO-generated Fortran program is called sj.f then the executable image is called sj (with no suffix).] For more details about compiling and linking, see section 1.2 of GPD-2 and section 6.3.1 in GPD-6.

When GEMSIM is used to carry out a simulation, the GEMSIM Auxiliary files (GSS and GST files) are required. They communicate vital information from the TABLO Input file to the simulation.

### 2.15.3 LOG Files

Often a program runs very quickly and/or puts a lot of information on the screen. If you need to check this output (for example, to see where an error occurs), you can ask for a LOG file to be created. You can then look at this log file in your favourite text editor. The suffix for LOG files is not prescribed by the software, though suffix **.LOG** is commonly used.[52]

### 2.15.4 Stored-input Files

If you have a lot of input to a program, or if you know you will need to run the program several times with the same, or similar, input, you can put all the input into a Stored-input file (see section 5.3). The suffix for Stored-input files is not prescribed by the software, though suffix **.STI** is commonly used. This is something you will need to learn about as you become more experienced. You can find introductory examples of creating and working with Stored-input files in sections 3.8.2, 3.8.4 and 4.3 below.

### 2.15.5 Files for Power Users

A number of files can be created in order to speed up or simplify subsequent tasks. These are typically used mainly by experienced power users of GEMPACK so you don't need to know any details about them at this stage. Examples are Stored-input files (see section 2.15.4 above), Environment files (see section 5.2 of GPD-3) and various Mapping files (see sections 8.3, 8.6 and 9.1 of GPD-4).

### 2.15.6 Work Files

Many programs create and use working files while they are running. These work files can be large. Usually these work files are deleted when the program finishes so you do not see them or need to know about them. Occasionally these files are left around on your hard disk if the program finishes with an error. [See section 12.7 of GPD-3 for more details.]

---

[52] On a Command file, you can use the statement "log file = yes ;" (see section 2.5.2 of GPD-3) to obtain a log file. All GEMPACK programs offer the option LOG at the start (see section 5.3.4).

### 2.15.7 Summary of Files

| File Type and Suffix | Input to | Output from |
|---|---|---|
| TABLO Input file (.TAB) | TABLO, TABmate | TABmate |
| Data files (often .DAT or .HAR) | Simulations, Data-manipulation TAB files, ViewHAR, MODHAR, SEEHAR etc | Simulations, Data-manipulation TAB files, ViewHAR, MODHAR etc |
| Command files (usually .CMF) | TABLO-generated programs, GEMSIM, SAGEM | (text editor) |
| Solution files (.SL4) | ViewSOL, GEMPIE, AnalyseGE, SLTOHT | TABLO-generated programs, GEMSIM, SAGEM |
| Equations files (.EQ4) | SAGEM, SUMEQ | TABLO-generated programs, GEMSIM |
| Shock files (often .SHK or .DAT) | TABLO-generated programs, GEMSIM, SAGEM | text editor, TABLO-generated programs, GEMSIM |
| SLC files (.SLC) | AnalyseGE | TABLO-generated programs and GEMSIM |
| Extrapolation Accuracy files (.XAC) | | TABLO-generated programs and GEMSIM |
| Auxiliary files (.GSS,.GST or .AXS, .AXT) | TABLO-generated programs, GEMSIM | TABLO |
| TABLO-generated Fortran program (.FOR or .f) | Compile and link – Step 1(b) in section 2.5.1 | TABLO |
| LOG files (usually .LOG) | | Any GEMPACK program |
| Stored-input files (usually .STI) | Any GEMPACK program | Text editor, or any GEMPACK program (see options SIF and ASI in section 5.3) |

**Table 2.15.7 : Summary of the Different Files**

## *2.16 For New Users - What Next?*

Congratulations. You have just learnt the most important things about GEMPACK, namely how to set up and carry out simulations, and how to look at the results.

Where you go next in learning about GEMPACK depends on your main purpose in using the software.

- If you mainly want to carry out simulations with another standard model, you will find a list of the models supplied with GEMPACK in section 1.8.
  If the model you wish to work with is one of these, you will find detailed hands-on guidance in GPD-8 about carrying out standard simulations with many of these models.
  If you wish to work with another model, the model developers have probably supplied and documented one or more standard simulations. We suggest that you start with those.
- With a well-documented model and good software, it is relatively easy to carry out simulations and produce, and report, a large number of results. It is less easy to really understand these results. The GEMPACK program AnalyseGE (see section 2.6 of GPD-4) provides valuable assistance to you in this task. You can find a hands-on introduction to AnalyseGE in the context of a Stylized Johansen simulation in section 6.1 of GPD-8. Pearson *et al* (2002) provides a hands-on introduction to AnalyseGE in the context of simulations with GTAP. We encourage you to explore the use of AnalyseGE as you analyse the results of simulations you carry out.
- If you want to build your own model (or modify someone else's), or if you just want to understand how a model is implemented in GEMPACK, you should read chapter 3. Perhaps read it quickly the first time and then go back for a more detailed study.
- If you need to build or modify the data files for a model, you should read chapter 4.
- When you want to know more about the GEMPACK utility programs (say, for report writing or post-simulation processing of results), look in GPD-4. GPD-4 contains detailed documentation about the GEMPACK utility programs.
- When you want to know more about running the GEMPACK programs, read chapter 5. This chapter gives detailed suggestions for more efficient use of the programs whether you are running them interactively or in batch mode.

Whatever you main interest, we strongly encourage you to at least skim chapter 3 first, and then go on to your main interest. You must be familiar with at least the basics of TABLO Input files in order to work properly with, and understand, any model implemented via GEMPACK.

# CHAPTER 3

# 3. Building or Modifying Models

In order to build a model within GEMPACK, it is necessary to prepare a TABLO Input file containing the equations of the model and to construct one or more data files whose purpose is essentially to give one solution of the levels equations of the model.

The preparation of a **TABLO Input file** involves

- writing down the equations in a suitable form. You can use levels equations, linearized equations or a mixture of these. We discuss this in section 3.1 below.
- working out the data requirements of the model. This is discussed in section 3.2 below.

We describe the preparation of TABLO Input files in section 3.3 and the preparation of the actual data files in chapter 4. We illustrate each step in the process by doing it for the Stylized Johansen model.

Of course, to modify an existing model, you modify the TABLO Input file (to change the theory of the model) and/or the data file(s).

The TABLO Input file given for Stylized Johansen in section 3.3.3 is a mixed one (in the sense that it contains a mixture of linearized and levels equations). In sections 3.4.1 and 3.5.1 we describe alternative TABLO Input files for Stylized Johansen consisting only of linearized or levels equations respectively. In section 3.6 we compare briefly the different sorts of representations - mixed, linearized or levels.

TABLO linearizes all levels equations in TABLO Input files and converts all levels variables to the associated linear ones (change or percentage change in the associated levels variables). This is described in section 3.7.

In section 3.8 we introduce the process of condensation. This can be used to reduce very large models to a manageable size.

We conclude this chapter in section 3.9 where we give you advice about building your own model by writing a TABLO Input file from scratch or by modifying an existing one. We include hands-on examples showing how to identify and correct errors in TABLO Input files and Command files.

If you are familiar with using GAMS for general equilibrium modelling, you may prefer to work through the document Kohlhaas and Pearson (2002) instead of, or before, reading this chapter. The many similarities between GEMPACK and GAMS make it relatively easy for a GAMS modeller to begin using GEMPACK productively.

| Levels Form* | Linearized Form | |
|---|---|---|
| *consumer demands*<br>$X_{i0} = \alpha_{i0} Y / P_i$ | $x_{i0} = y - p_i$ | i = 1,2 |
| *intermediate demands*<br>$X_{ij} = \alpha_{ij} X_j \prod_{t=1}^{4} P_t^{\alpha_{tj}} \left[ \prod_{t=1}^{4} (\alpha_{tj})^{-\alpha_{tj}} \right] / [A_j P_i]$ | $x_{ij} = x_j - (p_i - \sum_{t=1}^{4} \alpha_{tj} p_t)$ | i=1, ..., 4<br>j=1,2 |
| *price formation*<br>$P_j = \left[ \prod_{t=1}^{4} (\alpha_{tj})^{-\alpha_{tj}} \right] \prod_{t=1}^{4} P_t^{\alpha_{tj}} / A_j$ | $p_j = \sum_{t=1}^{4} \alpha_{tj} p_t$ | j = 1,2 |
| *commodity market clearing*<br>$\sum_{j=0}^{2} X_{ij} = X_i$ | $x_i = \sum_{j=0}^{2} [\frac{X_{ij}}{X_i}] x_{ij}$ | i = 1,2 |
| *aggregate primary factor usage*<br>$\sum_{j=1}^{2} X_{ij} = X_i$ | $x_i = \sum_{j=1}^{2} [\frac{X_{ij}}{X_i}] x_{ij}$ | i = 3,4 |
| *Numeraire*<br>$P_1 = 1$ | $p_1 = 0$ | |
| *Intermediate demands – dollar values*<br>$D_{ij} = P_i X_{ij}$ | $d_{ij} = p_i + x_{ij}$ | i=1, ..., 4<br>j=1,2 |
| *Consumer demands – dollar values*<br>$D_{i0} = P_i X_{i0}$ | $d_{i0} = p_i + x_{i0}$ | i = 1,2 |

* Upper-case Roman letters represent the levels of the variables; lower-case Roman letters are the corresponding percentage changes (which are the variables of the linearized version shown in the second column). The letters P, X and D denote prices, quantities and dollar values respectively, while the symbols A and $\alpha$ denote parameters. Subscripts 1 and 2 refer to the (single) commodities produced by industries 1 and 2 (subscript i), or to the industries themselves (subscript j); i = 3 refers to labour while i = 4 refers to the model's one (mobile-between-industries) type of capital; subscript j = 0 identifies consumption. Because the first three equation blocks are identically linear in the logarithms they are natural candidates for presentation and explanation of the model.

**Table 3.1.1a: Levels and Linearized Equations of the Stylized Johansen Model**

## *3.1 Writing Down the Equations of a Model*

TABLO Input files contain the equations of a model written down in a syntax which is very similar to ordinary algebra. Once you have written down the equations of your model in ordinary algebra, it is a simple matter to put them into a TABLO Input file, as we illustrate in section 3.3 below.

You are free to use levels or linearized versions of the equations or a mixture of these two types. For example, if a certain dollar value D is the product of the price P and quantity Q, the levels equation is

$$D = P * Q$$

(where the "*" indicates multiplication), and the associated linearized equation is

$$p\_D = p\_P + p\_Q$$

where "p_" denotes "percentage change in". The linearized version says that, to first order of approximation, the percentage-change in the dollar value is the sum of the percentage changes in the price and the quantity. Whichever version of the equation you include, GEMPACK can still produce accurate solutions of the underlying levels equations (which are usually nonlinear).

We say more about the process of linearizing equations in section 3.7 below.

The best way of making the above clear is to take a concrete example, as we do below, using Stylized Johansen as our example model.

### 3.1.1 Writing Down the Equations of Stylized Johansen

We start from the equations as written down in Chapter 3 of DPPW (to which we refer readers interested in the derivation of, and motivation behind, these equations).

The equations of the model are shown in Table 3.1.1a. In that table, both the levels and linearized versions of each equation are shown, taken essentially unchanged from DPPW.[53] Notice that, in Table 3.1.1a, upper case letters (for example, X) denote levels quantities while lower case letters (for example, x) denote percentage change in the corresponding levels quantity.

For our first implementation of Stylized Johansen (see section 3.3 below), we have chosen a mixed representation, based on the shaded blocks in Table 3.1.1a. That is, we decided to use the levels versions of some of the equations (most are accounting identities and one is the numeraire equation) and the linearized versions of the top three equations (which are behavioural equations). Later, in sections 3.4 and 3.5 respectively, we describe implementations based on exclusively linearized equations (section 3.4) and exclusively levels equations (section 3.5). Of course, each of these 3 implementations is valid and all three produce the same results.

The notation in DPPW involves a liberal use of subscripts which are not suitable for the linear type of input usually required by computers (and required in the TABLO Input file). Hence we use a different notation from DPPW. The levels variables of the model are as follows. In DPPW subscripts 1 and 2 refer to sectors (commodity or industry), subscripts 3 and 4 refer to factors (3 is labor and 4 is capital) while subscript 0 refers to households.

---

[53] The last 2 rows in Table 3.1.1a, which relate dollar values to prices and quantities, are not explicitly written down in DPPW but, of course, underlie the treatment there. The levels equations are (E3.1.9) [consumer demands], (E3.1.10), (E3.1.12), (E3.1.6), (E3.1.7) and (E3.1.23) [numeraire] in DPPW, while the corresponding linearized equations are (E3.2.1), (E3.2.2), (E3.2.3), (E3.2.4), (E3.2.5) and (E3.2.6) respectively.

**Table 3.1.1b: Levels Variables for Stylized Johansen**

| GEMPACK variable | Meaning | DPPW Notation |
|---|---|---|
| Y | Value of household income | Y |
| PC(i) | Price of commodity i | $P_i$ (i=1,2) |
| PF(f) | Price of factor f | $P_f$ (f=3,4) |
| XCOM(i) | Supply of commodity i | $X_i$ (i=1,2) |
| XFAC(f) | Supply of factor f | $X_f$ (f=3,4) |
| XH(i) | Household use of commodity i | $X_{i0}$ (i=1,2) |
| XC(i,j) | Intermediate input of commodity i to industry j | $X_{ij}$ (i,j=1,2) |
| XF(f,j) | Input of factor f to industry j | $X_{fj}$ (f=3,4;j=1,2) |
| DVCOMIN(i,j) | Dollar values for intermediate inputs | (i,j=1,2) |
| DVFACIN(f,j) | Dollar values for factor use by industry | (f=3,4;j=1,2) |
| DVHOUS(i) | Dollar values for household consumption | (i=1,2) |

**Table 3.1.1c: Parameters for Stylized Johansen Levels Equations**

| Parameters | | DPPW Notation |
|---|---|---|
| ALPHACOM(i,j) | Commodity exponents in production function for sector j (E3.1.4) | $ALPHA_{ij}$ (i,j=1,2) |
| ALPHAFAC(i,j) | Factor exponents in production function for sector j (E3.1.4) | $ALPHA_{fj}$ (f=3,4; j=1,2) |

We introduce sets SECT, the set of two sectors say "s1" and "s2", and FAC, the set of the two factors "labor" and "capital".

Below in Table 3.1.1d, we have rewritten the selected equations from Table 3.1.1a, this time using the GEMPACK variables and notation as in Tables 3.1.1b and 3.1.1c. Note that below we also use the GEMPACK convention that "p_" indicates percentage change in the relevant levels variable. For example, p_XH(i) denotes the percentage change in XH(i), household consumption of commodity i. In these equations we use "*" to denote multiplication and "/" to denote division. We also use SUM(i,<set>,<expression>) to denote sums (usually expressed via Greek sigma) over all i in the set <set>; here <set> is SECT or FAC.

**Table 3.1.1d: Stylized Johansen Equations in GEMPACK notation**

```
(E1) p_XH(i) = p_Y - p_PC(i)                i in SECT
```
[This is equation (E3.2.1) in DPPW]

```
(E2) p_XC(i,j) = p_XCOM(j) - [p_PC(i) - p_PC(j)]   i,j in SECT
```
[This is obtained from equation (E3.2.2) for i=1,2 in DPPW. The term p_PC(j) is included because of equation (E3.2.3) in DPPW.]

```
(E3) p_XF(f,j) = p_XCOM(j) - [p_PF(f) - p_PC(j)]   f in FAC,j in SECT
```
[This is obtained from equation (E3.2.2) for i=3,4 in DPPW. The term p_PC(j) is included because of equation (E3.2.3) in DPPW.]

```
(E4) p_PC(j) = SUM(i,SECT, ALPHACOM(i,j)*p_PC(i)) +
               SUM(f,FAC,  ALPHAFAC(f,j)*p_PF(f))          j in SECT
```
[This is equation (E3.2.3) in DPPW]

```
(E5) XCOM(i) = XH(i) + SUM(j,SECT, XC(i,j))        i in SECT
```
[This is equation (E3.1.6) in DPPW]

```
(E6) XFAC(f) = SUM(j,SECT, XF(f,j))                f in FAC
```
[This is equation (E3.1.7) in DPPW]

```
(E7) PC("s1") = 1
```
[This is equation (E3.1.23) in DPPW]

```
(E8) XC(i,j) = DVCOMIN(i,j) / PC(i)                i,j in SECT
```
[This equation is not numbered in DPPW]

```
(E9) XH(i)   = DVHOUS(i) / PC(i)                   i in SECT
```
[This equation is not numbered in DPPW]

```
(E10) XF(f,j) = DVFACIN(f,j) / PF(f)               f in FAC, j in SECT
```
[This equation is not numbered in DPPW]

These equations appear essentially as above in the TABLO Input file (see section 3.3.3 below).

## *3.2 Data Requirements for the Linearized Equations*

As a general rule, GEMPACK requires an initial levels solution of the model. Thus it is necessary to provide data from which initial (that is, pre-simulation) values of all levels variables and the values of all parameters of the model can be inferred.

As we shall see in section 3.2.1 for Stylized Johansen, it is frequently the case that the data required are

- mainly dollar values (rather than separate prices and quantities), and
- certain parameters (such as elasticities).

Once dollar values are known, it is often possible to set basic prices equal to 1 (this amounts to a choice of units for the related quantities), from which the quantities can be derived by dividing the dollar value by the price. [The choice of 1 for the basic price is, of course, arbitrary. Any other fixed value would be as good.]

### 3.2.1 Data Requirements for Stylized Johansen

Suppose that we know the following pre-simulation dollar values.

| | |
|---|---|
| `DVCOMIN(i,j)` | Intermediate inputs |
| `DVHOUS(i)` | Household consumption |
| `DVFACIN(f,j)` | Factor use by industry |

Then, if we set all the prices to one for

| | |
|---|---|
| `PC(i)` | Price of commodities |
| `PF(f)` | Price of factors |

we can infer all other levels variables in Table 3.1.1b as follows.

| | |
|---|---|
| `XC(i,j) = DVCOMIN(i,j) / PC(i)` | Intermediate inputs |
| `XH(i)  = DVHOUS(i) / PC(i)` | Household use |
| `XF(f,j) = DVFACIN(i,j) / PF(f)` | Factor use |
| `Y  =  SUM(i, SECT, DVHOUS(i))` | Household expenditure |

The only other quantities in the equations (E1)-(E10) are the parameters ALPHACOM(i,j) and ALPHAFAC(f,j) in (E4). Because there is a Cobb-Douglas production function involved, it is well-known that these are cost shares, namely

```
ALPHACOM(i,j) = DVCOMIN(i,j) / DVCOSTS(j),
ALPHAFAC(f,j) = DVFACIN(f,j) / DVCOSTS(j),
```

where DVCOSTS(j) is an abbreviation for

```
SUM(i, SECT, DVCOMIN(i,j)) + SUM(f, FAC,  DVFACIN(f,j)),
```

the total costs in industry j. [These results are easily obtained from equations (E3.1.10) and (E3.1.12) in DPPW.]

Thus the only data requirements are the dollar values

`DVHOUS(i),  DVCOMIN(i,j)  and  DVFACIN(f,j).`

One instance of the data required is as shown in the body of Table 2.1.1a in section 2.1.1 above.

In the TABLO Input file, the pre-simulation values of these data will be read and the values of all others will be calculated from them.

## *3.3 Constructing the TABLO Input File for a Model*

The TABLO Input file of the model is the means of communicating the theory of the model to the computer, in particular to the GEMPACK program TABLO. It consists of the equations written in a syntax which is very similar to ordinary algebra. It also contains a description of the data to be read, where it is to be read from, and how this data is to be used to calculate values of parameters and pre-simulation values of the other levels variables occurring in the equations.

The main part of a TABLO Input file is the equations, which usually come at the end of the file. Before the equations must come

- the VARIABLEs (levels or linearized) occurring;
- the SETs used to describe the different arguments of variables;
- the data to be read;
- calculations of the pre-simulation values of any levels variables not read in as data (calculations are done via FORMULAs);
- calculations (via FORMULAs) of any parameters whose values are not read in;
- logical names of the associated data files;
- the headers on the data file(s) where the different pieces of data are to be found (if the data files are GEMPACK Header Array files - see section 4.1.1 below).

The order of these in the TABLO Input file is somewhat flexible but follows the general rule that items cannot be used until they have been declared. Thus the SET statements (saying which sets are involved) usually come first. Then the declarations of data files (via FILE statements) often come next, followed by the declarations of the VARIABLEs and parameters.

These ideas are best learnt and understood by example. Hence we launch straight into the preparation of the TABLO Input file for Stylized Johansen.

### 3.3.1 Viewing the TABLO Input File

When working with files in GEMPACK, many of the files you work on and create are text files, so you need a text editor.[54] There are various text editors available under Unix. You can use any text editor you are familiar with (for example, vi).

If you are working on a Windows PC, there are two text editors called TABmate and GemEdit

> supplied with the Windows version of GEMPACK. We suggest you use **TABmate** since it has coloured highlighting of TABLO syntax, and a powerful Gloss feature which displays all parts of the TABLO code where a chosen variable or coefficient is used. If TABmate is not the default editor in WinGEM, select, in the WinGEM main menu,
>
> ***Options | Change editor...***
>
> then select your editor ***Use TABmate.*** You should only have to do this once, since WinGEM should remember which editor you have chosen.
>
> Open the TABLO Input file for Stylized Johansen SJ.TAB in the TABmate editor. In WinGEM, to edit a text file, select in the WinGEM menu,

[54] You could use a word processor but then you would always have to remember to save the file as a text or ASCII file.

***File | Edit file...***

The edit box should show various files in the directory C:\SJ. If the Open box does not start at the correct directory C:\SJ, set the Working directory again as described in section 2.4.3.

Select the file **SJ.TAB**, the TABLO Input file for the Stylized Johansen model.

Search the TABLO Input file for "EQUATION House" using ***Search | Find***. We will discuss the details of this equation in the next section.

### 3.3.2 Constructing Part of the TABLO Input File for Stylized Johansen

In this subsection we consider just two equations of Stylized Johansen, namely (E9) and (E4) in section 3.1.1 above. We show how these are written in the TABLO Input file. (We show the full TABLO Input file in section 3.3.3 and then discuss the rest of this file in section 3.3.4 below.)

**Equation (E9)**

Consider the very simple equation (E9) relating prices, quantities and dollar values of household consumption.

In the TABLO Input file this equation is written as[55]

```
EQUATION House # Household demand for commodity i #
    (all,i,SECT) XH(i) = DVHOUS(i) / PC(i) ;
```

where

- **`EQUATION`** is a keyword indicating that what follows is an equation,
- **`House`** is the name by which this equation is known in the model,
- the words between the hashes **#** form optional additional labelling information which is associated with the equation,
- the quantifier **`(all,i,SECT)`** indicates that there are really several equations, one for each sector, and
- the semicolon **`;`** marks the end of this part of the equation statement.

For this equation to be meaningful, we must explain in the TABLO Input file all the names used in the equation.

The levels variables are declared (that is, explained) via the statements

```
VARIABLE (all,i,SECT) XH(i)
      # Household demand for commodity i # ;
VARIABLE (all,i,SECT) DVHOUS(i)
      # Dollar value of household use of commodity i # ;
VARIABLE (all,i,SECT) PC(i) # Price of commodity i # ;
```

Notice that, by convention, these declarations also declare associated linear variables p_XH, p_DVHOUS and p_PC which denote the percentage-change in the relevant levels variables. These linear variable names are used in reporting simulation results (see the results in section 2.7 above, for example) and are available for use in linearized equations in the TABLO Input file (see, for example, the EQUATION named "Price_formation" discussed later in this section) without further explicit declaration.

The fact that SECT is a set with two sectors "s1" and "s2" in it is indicated via the SET statement

---

[55] The reason for writing XH(i)=DVHOUS(i)/PC(i) rather than DVHOUS(i)=PC(i)*XH(i) will become clear when we discuss identifying pre-simulation values.

```
SET SECT  # Sectors #   (s1-s2) ;
```

We must also indicate how pre-simulation values of the levels variables can be read or calculated from the data base. We do this via the statements

```
READ DVHOUS  from FILE iodata HEADER "HCON" ;
FORMULA (all,i,SECT)   PC(i) = 1 ;
FORMULA (all,i,SECT)   XH(i) = DVHOUS(i)/PC(i) ;
```

In the first of the above statements,

- **READ** is the keyword,
- **iodata** is the (logical) name by which the particular data file containing this input-output data is known in the TABLO Input file, and
- the Header "HCON" tells where on the file the relevant array of data is to be found.

In the second and third statements, **FORMULA** is the keyword.

The third of these contains the same expression as the equation we are considering. Indeed, we can combine the EQUATION and FORMULA into a single statement on the TABLO Input file, namely[56]

```
FORMULA & EQUATION House # Household demand for commodity i #
  (all,i,SECT)   XH(i) = DVHOUS(i) / PC(i) ;
```

The statement

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

declares "iodata" as the logical name[57] of the file containing the actual data.

This ends the discussion of the equation (E9) and of all the statements needed for the EQUATION House.

**Using the Gloss Feature in TABmate (Windows PCs only)**

In TABmate, there is a quick way of finding all places where a name such as XH occurs in the TABLO Input file SJ.TAB. In TABmate, click on ***TABLO Check*** in the bar near the top of the TABmate screen. Click on the word EQUATION in the "EQUATION House" and then click on ***Gloss*** in the bar near the top of the TABmate screen. A box (shown below) appears showing all names used in the EQUATION House in the TABLO Input file.

```
FORMULA & EQUATION House # Household demand for commodity i # (all,i,SECT) XH(i) = DVHOUS(i) / PC(i) ;

~41: SET SECT # Sectors # (s1-s2) ;
~69: Coefficient Variable (GE 0) (all,i,SECT) XH(i) # Household demand for commodity i # ;
~86: Coefficient Variable (GE 0) (all,i,SECT) DVHOUS(i) # Dollar value of household use of commodity i # ;
~59: Coefficient Variable (GE 0) (all,i,SECT) PC(i) # Price of commodity i # ;
```

Click on the X in the top right-hand corner of the Glossary box to exit from the Gloss box.

Click on the word **XH** in the "EQUATION House" and then click on the ***Gloss*** button. The Gloss box (shown below) appears showing all occurrences of the name XH in the TABLO Input file.

```
Coefficient   XH
  Line
~69: Variable (GE 0) (all,i,SECT) XH(i) # Household demand for commodity i # ;
~138: FORMULA & EQUATION House # Household demand for commodity i # (all,i,SECT) XH(i) = DVHOUS(i) / PC(i) ;
~142: FORMULA & EQUATION Com_clear #Commodity market clearing # (all,i,SECT) XCOM(i) = XH(i)+SUM(j,SECT,XC(i,j));
```

---

[56] This explains why we have written the equation as shown rather than the more natural DVHOUS(i)=PC(i)*XH(i).

[57] The actual name of this file on your computer can be quite different from this logical name which is just used in the TABLO Input file to distinguish between possibly several different logical files.

Try clicking on the word DVHOUS to see where DVHOUS is used in the TABLO Input file SJ.TAB.

**Equation (E4)**

Now consider the equation (E4) "price formation for commodities". This is written in the TABLO Input file as

```
EQUATION (LINEAR) Price_formation
(all,j,SECT) p_PC(j) = SUM(i,SECT, ALPHACOM(i,j)*p_PC(i)) +
                               SUM(f,FAC, ALPHAFAC(f,j)*p_PF(f)) ;
```

in which

- the qualifier **(LINEAR)** indicates that this is a linearized equation (not a levels equation),
- the fact that p_PC(i) and p_PF(f) are percentage-changes in the levels variables PC(i) and PF(f) is guaranteed by the convention that, once these levels variables have been declared via

```
VARIABLE (all,i,SECT) PC(i) # Price of commodity i # ;
VARIABLE (all,f,FAC)  PF(f) # Price of factor f # ;
```

the associated linear variables p_PC(i) and p_PF(f) are automatically considered declared. In this equation, ALPHACOM and ALPHAFAC are parameters. To calculate ALPHACOM and ALPHAFAC from the data base, FORMULA statements are used:

```
FORMULA
# Share of intermediate commodity i in costs of industry j #
(all,i,SECT)(all,j,SECT)  ALPHACOM(i,j) = DVCOMIN(i,j) /
    [SUM(ii,SECT,DVCOMIN(ii,j)) + SUM(ff,FAC,DVFACIN(ff,j)) ] ;
FORMULA
# Share of factor input f in costs of industry j #
(all,f,FAC)(all,j,SECT)  ALPHAFAC(f,j) = DVFACIN(f,j) /
    [SUM(ii,SECT,DVCOMIN(ii,j)) + SUM(ff,FAC,DVFACIN(ff,j)) ] ;
```

where **FORMULA** is the keyword. The fact that ALPHACOM and ALPHAFAC are parameters can be indicated via the statements

```
COEFFICIENT(PARAMETER) (all,i,SECT)(all,j,SECT) ALPHACOM(i,j) ;
COEFFICIENT(PARAMETER) (all,f,FAC) (all,j,SECT) ALPHAFAC(f,j) ;
```

in which **COEFFICIENT** is the keyword and **(PARAMETER)** is a qualifier.

If you are using the TABmate editor, try using the ***Gloss*** button. Click on the word **p_PC** in the EQUATION Price_formation and click on the Gloss button to see all occurrences of p_PC in the TABLO Input file SJ.TAB. Click on the word "FAC" and ***Gloss*** to see all occurrences of the set FAC.

This ends the discussion of the equation (E4) and of all the statements needed for the EQUATION Price_formation.

**Statements in a TABLO Input File**

The main types of statements in a TABLO Input file, namely EQUATIONs, FORMULAs, READs, VARIABLEs, COEFFICIENTs, SETs and FILEs have been introduced in connection with equations House (E9) and Price_formation (E4) above.

In addition, if you want to check the values of say ALPHAFAC, you could add one of the statements

```
DISPLAY  ALPHAFAC ;
WRITE ALPHAFAC TO TERMINAL ;
WRITE  ALPHAFAC TO FILE xxx ;
```

(where "xxx" would need to be declared as a FILE).

Here **DISPLAY** and **WRITE** are the keywords (see sections 3.12 and 3.7 of GPD-2, and chapter 4 of GPD-3 for details). These statements can be added anywhere after the FORMULA giving the values of ALPHAFAC.

Each entity (VARIABLE, COEFFICIENT, etc) must be declared in the TABLO Input file before it is used in EQUATIONs and FORMULAs. This partly determines the order of the statements in the TABLO Input file.

We suggest that you now look at the complete TABLO Input file for this model, as set out in section 3.3.3 below, or using the editor TABmate (or your text editor). This file is usually called SJ.TAB when supplied with the GEMPACK Examples. You will find all the statements shown above (except the DISPLAY and WRITE statements) in that file.

Since declarations must come before use, you will find the TABLO statements in pretty much the reverse order from that in which we have introduced them above.

We discuss the rest of this TABLO Input file in section 3.3.4.

### 3.3.3 The TABLO Input File for the Stylized Johansen Model[58]

```
!------------------------------------------------------------------!
!                  Mixed TABLO Input file for the                  !
!                    Stylized Johansen model                       !
!                                                                  !
!         following the description in Chapter 3 of the text       !
!   "Notes and Problems in Applied General Equilibrium Economics"  !
!       by P.Dixon, B.Parmenter, A.Powell and P.Wilcoxen [DPPW]    !
!              published by North-Holland 1992.                    !
!                                                                  !
!------------------------------------------------------------------!
! Text between exclamation marks is a comment.                     !
! Text between hashes (#) is labelling information.                !
!------------------------------------------------------------------!
!     Set default values                                           !
!------------------------------------------------------------------!
VARIABLE (DEFAULT = LEVELS) ;
EQUATION (DEFAULT = LEVELS) ;
COEFFICIENT (DEFAULT = PARAMETER) ;
FORMULA (DEFAULT = INITIAL) ;
!------------------------------------------------------------------!
!      Sets                                                        !
!------------------------------------------------------------------!
! Index values i=1,2 in DPPW correspond to the sectors called s1,s2.
  Index values i=3,4 in DPPW correspond to the primary factors,
  labor and capital.   The set SECT below doubles as the set of
  commodities  and the set of industries. !
SET SECT # Sectors # (s1-s2)  ;
SET FAC # Factors  # (labor, capital) ;
SET NUM_SECT # Numeraire sector - sector 1 #  (s1) ;
SUBSET  NUM_SECT is subset of SECT ;
```

[58] If you look closely at the file SJ.TAB supplied with GEMPACK, you will see that all the Variable declarations have a qualifier "(GE 0)" which is not shown in the version of SJ.TAB printed here. The purpose of these is to tell the software that these Variables must never become negative (in the levels). These qualifiers only affect simulation results in certain circumstances which are a little too complicated to go into at this stage of our documentation. We refer interested readers to section 6.4 of GPD-3.

```
!-----------------------------------------------------------------------!
!      Levels variables                                                 !
!-----------------------------------------------------------------------!
! In the DPPW names shown below, : denotes subscript.                   !
! For example, x:j indicates that j is a subscript.                     !
VARIABLE               Y    # Total nominal household expenditure #
                              ! This is also Y in DPPW ! ;
VARIABLE (all,i,SECT)  PC(i)   # Price of commodity i #
                               ! This is p:i (i=1,2) in DPPW ! ;
VARIABLE (all,f,FAC)   PF(f)   # Price of factor f #
                               ! This is p:i (i=3,4) in DPPW ! ;
VARIABLE (all,i,SECT)  XCOM(i)   ! This is x:i (i=1,2) in DPPW !
       # Total demand for (or supply of) commodity i # ;
VARIABLE (all,f,FAC)   XFAC(f)   ! This is x:i (i=3,4) in DPPW !
      # Total demand for (or supply of) factor f    # ;
VARIABLE (all,i,SECT)  XH(i)   # Household demand for commodity i #
                               ! This is x:i0 (i=1,2) in DPPW ! ;
VARIABLE (all,i,SECT) (all,j,SECT)     XC(i,j)
      # Intermediate inputs of commodity i to industry j #
     ! This is x:ij (i,j=1,2) in DPPW ! ;
VARIABLE (all,f,FAC)(all,j,SECT)       XF(f,j)
      # Factor inputs to industry j #
     ! This is x:ij (i=3,4; j=1,2) in DPPW ! ;
!-----------------------------------------------------------------------!
!     Dollar values read in from database                               !
!-----------------------------------------------------------------------!
VARIABLE (all,i,SECT)(all,j,SECT)      DVCOMIN(i,j)
      # Dollar value of inputs of commodity i to industry j # ;
VARIABLE (all,f,FAC)(all,j,SECT)       DVFACIN(f,j)
     # Dollar value of factor f used in industry j # ;
VARIABLE (all,i,SECT)                  DVHOUS(i)
      # Dollar value of household use of commodity i # ;
!-----------------------------------------------------------------------!
!     Parameters                                                        !
!-----------------------------------------------------------------------!
COEFFICIENT (all,i,SECT)(all,j,SECT)  ALPHACOM(i,j)
   # Share of intermediate use of commodity i in costs of industry j # ;
COEFFICIENT (all,f,FAC)(all,j,SECT)   ALPHAFAC(f,j)
   # Share of factor input f in costs of industry j # ;
!-----------------------------------------------------------------------!
!      File                                                             !
!-----------------------------------------------------------------------!
FILE iodata # input-output data for the model # ;
!-----------------------------------------------------------------------!
!      Reads from the data base                                         !
!-----------------------------------------------------------------------!
READ DVCOMIN from FILE iodata HEADER "CINP" ;
READ DVFACIN from FILE iodata HEADER "FINP" ;
READ DVHOUS  from FILE iodata HEADER "HCON" ;
!-----------------------------------------------------------------------!
!      Formulas                                                         !
!-----------------------------------------------------------------------!
FORMULA (all,i,SECT) PC(i) = 1.0 ;
FORMULA (all,i,FAC) PF(i) = 1.0 ;
FORMULA (all,i,SECT)(all,j,SECT) ALPHACOM(i,j) = DVCOMIN(i,j) /
            [SUM(ii,SECT,DVCOMIN(ii,j)) + SUM (ff,FAC,DVFACIN(ff,j))] ;
FORMULA (all,f,FAC)(all,j,SECT) ALPHAFAC(f,j) = DVFACIN(f,j) /
            [SUM(ii,SECT,DVCOMIN(ii,j)) + SUM (ff,FAC,DVFACIN(ff,j))] ;

! Formula to give initial value of Y !
FORMULA Y = SUM(i,SECT,DVHOUS(i)) ;
```

```
!-------------------------------------------------------------------!
!       Formulas and levels equations                               !
!-------------------------------------------------------------------!
FORMULA & EQUATION Comin
     # Intermediate input of commodity i to industry j #
   (all,i,SECT)(all,j,SECT) XC(i,j) = DVCOMIN(i,j) / PC(i) ;
FORMULA & EQUATION Facin      # Factor input f to industry j #
 (all,f,FAC)(all,j,SECT) XF(f,j) = DVFACIN(f,j) / PF(f) ;
FORMULA & EQUATION House      # Household demand for commodity i #
 (all,i,SECT) XH(i) = DVHOUS(i) / PC(i) ;
FORMULA & EQUATION Com_clear ! (E3.1.6) in DPPW !
    # Commodity market clearing #
 (all,i,SECT) XCOM(i) = XH(i) + SUM(j,SECT,XC(i,j)) ;
FORMULA & EQUATION Factor_use ! (E3.1.7) in DPPW !
     # Aggregate primary factor usage #
 (all,f,FAC) XFAC(f) = SUM(j,SECT,XF(f,j)) ;
!-------------------------------------------------------------------!
!      Equations                                                    !
!-------------------------------------------------------------------!
EQUATION(LINEAR)  Consumer_demands ! (E3.2.1) in DPPW !
     # Household expenditure functions #
 (all,i,SECT) p_XH(i) = p_Y - p_PC(i) ;

EQUATION(LINEAR)  Intermediate_com
! From (E3.2.2) with i=1,2 in DPPW. The term p_PC(j) is included
   because of (E3.2.3) in DPPW. !
    # Intermediate demands for commodity i by industry j #
(all,i,SECT)(all,j,SECT) p_XC(i,j) = p_XCOM(j) - (p_PC(i) - p_PC(j)) ;

EQUATION(LINEAR)  Factor_inputs
! From (E3.2.2) with i=3,4 in DPPW. The term p_PC(j) is included
   because of (E3.2.3) in DPPW. !
    # Factor input demand functions #
(all,f,FAC)(all,j,SECT) p_XF(f,j) = p_XCOM(j) - (p_PF(f) - p_PC(j)) ;

EQUATION(LINEAR) Price_formation   ! (E3.2.3) in DPPW !
     # Unit cost index for industry j #
 (all,j,SECT) p_PC(j) = SUM(i,SECT,ALPHACOM(i,j)*p_PC(i)) +
                              SUM(f,FAC,ALPHAFAC(f,j)*p_PF(f)) ;

EQUATION Numeraire  ! (E3.1.23) in DPPW !
     # Price of commodity 1 is the numeraire #
 (all,i,NUM_SECT)  PC(i) = 1 ;

!--------------end of TABLO Input file-----------------------------!
```

### 3.3.4 Completing the TABLO Input File for Stylized Johansen

Notice that the TABLO Input file consists of a number of **statements**, each beginning with its relevant **keyword** (such as SET or VARIABLE). Some statements include a **qualifier** such as (LINEAR) in EQUATION(LINEAR). Each statement ends with a semicolon ';'. Text between exclamation marks '!' is treated as a **comment**; such text can go anywhere in the TABLO Input file. Text between hashes '#' is **labelling information**; the positioning of this labelling information is restricted (see chapter 3 of GPD-2 for full details).

The TABLO Input file is **not case-sensitive** so for example, XH and Xh would be identical so far as TABLO is concerned.

First come the **DEFAULT** statements. In TABLO Input files, EQUATIONs and VARIABLEs can be linear or levels. It is possible to distinguish each type by using the appropriate qualifier (LEVELS) or (LINEAR) after the keyword each time, as in, for example,

```
VARIABLE (LEVELS) Y # Nominal household expenditure # ;
VARIABLE (LINEAR) (all,f,FAC) p_PF(f) # Price of factors # ;
```

When most variables being declared are levels variables, it seems wasteful to have to keep repeating the qualifier (LEVELS). There are DEFAULT statements which allow you to reduce the number of qualifiers required in your TABLO Input files. If you put the statement

```
VARIABLE (DEFAULT = LEVELS) ;
```

early in a TABLO Input file, then, after it, any VARIABLE declaration is taken as the declaration of a levels variable unless a different qualifier (LINEAR) is present. Similarly for EQUATIONs coming after the statement

```
EQUATION (DEFAULT = LEVELS) ;
```

Of course, if most equations in your TABLO Input file are linearized ones, you could put the opposite default statement

```
EQUATION (DEFAULT = LINEAR) ;
```

near the start of your file, and then you would only have to flag, using the qualifier (LEVELS), the levels equations.

Similarly, the statements

```
COEFFICIENT (DEFAULT = PARAMETER) ;
FORMULA (DEFAULT = INITIAL) ;
```

set the default types for COEFFICIENTs declared and FORMULAs. The only COEFFICIENTs in the TABLO Input file in section 3.3.3 above are parameters, while the only FORMULAs are used to set initial values (that is, pre-simulation values) of levels variables, or to set the values of the parameters. You will see non-parameter COEFFICIENTs and non-initial FORMULAs in section 3.4.1 below, when you look at linearized TABLO Input files.

Next come the declarations of the **SET**s, namely SECT (sectors) and FAC (primary factors). A further set NUM_SECT to stand for the single numeraire sector (sector s1) is also defined; this is only used for the last of the equations, the numeraire equation. The reason for the SUBSET statement will be explained when we discuss that equation below.

Then come the declarations of the **VARIABLE**s. Note that the arguments (if any) of each are clearly described, using the "(all,<index>,<set-name>)" **quantifier**(s) at the start of the declarations.

These quantifiers refer to the SETs, which is why the SET declarations must precede the VARIABLE declarations. The variables declared are all levels variables (because of the DEFAULT statement earlier). Although not explicitly mentioned here, the associated linear variables p_Y, p_XH etc are

taken as automatically declared by convention, and can be used in subsequent EQUATIONs without further explicit declaration.

Then comes the declaration of the parameters - which must always be declared as COEFFICIENTs. The qualifier (PARAMETER) is not needed here because of the earlier DEFAULT(COEFFICIENT=PARAMETER) statement.

Next comes the declaration of the single data **FILE** required. This file is given the logical name 'iodata'. The actual name of the file on your computer containing this data is not limited by this logical name. You can give the actual file any convenient name. GEMSIM or the TABLO-generated program will prompt you for this actual name when you run it; the prompt will use the logical name 'iodata' from the TABLO Input file. Or, if you use a GEMPACK Command file (as we recommend), you will need to use the logical name as well as the actual name in the relevant statement (for example, "file iodata = sj.dat ;"). See section 4.1 of GPD-3 for more details.

Then come **READ** statements telling the program to read in initial (that is, pre-simulation) values of certain levels variables. Each READ statement says from where the data is to be read (that is, which file and which header on the file).

Next come some FORMULAs assigning initial values to other levels variables. The left-hand side of a FORMULA (that is, the part before the '=' sign) must be a simple VARIABLE or COEFFICIENT, but the right-hand side can be a complicated expression. In such an expression, the symbols for the arithmetic operations are '+' and '-' for addition and subtraction, '*' and '/' for multiplication and division, and '^' for exponentiation. Note that '*' must be shown explicitly wherever multiplication is required. Notice also the use of the syntax

SUM(<index>,<set-name>, <expression to be summed> )

to express sums over sets.

Finally come the **EQUATION**s (see (E1) to (E10) in section 3.1.1 above). As explained in section 3.3.2, some of these double as FORMULAs, in which case the statement must begin with **FORMULA & EQUATION** to indicate that there are really two statements here.

The syntax of the last equation (the numeraire equation) may surprise you. We could have expressed this as

```
PC("s1") = 1 ;
```

using the sector element name "s1" to indicate which price is fixed at one. Instead we have introduced the new set NUM_SECT consisting of just this sector "s1" and written the equation as

```
(all,i,NUM_SECT)  PC(i) = 1 ;
```

This illustrates the point of SUBSET declarations. The VARIABLE PC has been declared to have one argument ranging over the set SECT, but here we need to give it an argument ranging over the smaller set NUM_SECT. The earlier SUBSET statement

```
SUBSET  NUM_SECT  is subset of  SECT ;
```

alerts TABLO to the fact that an argument ranging over NUM_SECT is always in the set SECT. Without this, the use of PC(i) with i ranging over NUM_SECT would trigger a semantic error since TABLO checks that all arguments range over appropriate sets.

As stated earlier, the order of the statements in the TABLO Input file can be varied. For example, especially with larger models, some COEFFICIENTs may only be relevant to a small number of the EQUATIONs and it may be better to declare these and assign values to them just before the relevant EQUATION or group of EQUATIONs.

Note also that there are DISPLAY and WRITE statements to enable you to look at the values of COEFFICIENTs (or levels VARIABLEs) as calculated and/or to write other files (text or Header

Array files) via GEMSIM or TABLO-generated programs. [These features give TABLO some of the properties of a data base manipulator, as explained in section 4.4 below.] You might like to try them out by adding the following statements at the end of the TABLO Input file for Stylized Johansen and then re-running Steps 1,2 and 3 in chapter 2.

```
DISPLAY ALPHACOM ;
WRITE ALPHAFAC TO TERMINAL ;
FILE (NEW, TEXT) output ;
WRITE ALPHACOM TO FILE output ;
WRITE ALPHAFAC TO FILE output ;
```

Complete documentation of TABLO Input files is given in **GPD-2** *TABLO Reference*, which you will need to consult when you start to build a new model.

### 3.3.5 Change or Percentage-Change Variables

Many levels variables (for example, prices, quantities, dollar values) are always positive and it is usual to work with the associated percentage change as the associated linear VARIABLE.

However, when the relevant levels variable can be positive or zero or negative (examples are the Balance of Trade and an *ad valorem* tax rate), it is wiser to work with the associated **change** as a linear VARIABLE. This is because, in such a case, if the levels value happens to be zero at the start of any step of a multi-step simulation, the associated percentage change could not be calculated (since it would require division by zero). Also, there are often numerical problems (which slow or hinder convergence of the solutions) when a percentage-change variable changes sign in the levels[59]; these problems may be overcome if a change variable is used because then TABLO often works with a slightly different linearization of the EQUATIONs involving this VARIABLE.[60]

In summary, we suggest the following guidelines.

- For levels variables which are always positive (or always negative), work with the associated percentage change as a linear VARIABLE.
- For levels variables which may be positive, zero or negative, work with the associated change as a linear VARIABLE. In this case, if you are declaring the levels variable, insert the qualifier **(CHANGE)** after the keyword VARIABLE. This tells TABLO to automatically declare the associated change as a linear variable (the prefix "**c_**" is usually added to the levels name[61]). For example, if you have a declaration

  ```
  VARIABLE (CHANGE) BT  # Balance of trade # ;
  ```

  in your TABLO Input file, the associated change linear variable c_BT is automatically available for use in linearized equations and will be used in reporting simulation results.
  Alternatively a linear change variable can be declared directly, using the two qualifiers LINEAR and CHANGE as in

  ```
  VARIABLE (LINEAR,CHANGE) delB  # Change in trade balance # ;
  ```

  (When you declare a linear change variable explicitly, you are not required to begin the name with "c_".)

---

[59] See section 12.6.4 of GPD-3 for an example.

[60] See section 2.2.6 of GPD-2.

[61] See section 2.2.2 of GPD-2 for details.

### 3.3.6 Variable or Parameter ?

When you build a model, you have in mind the sorts of questions you will be using the model to answer. You may be thinking of holding some quantities constant and varying others.

Within GEMPACK, the quantities you may wish to vary will be declared as VARIABLEs while those which cannot vary can be declared as COEFFICIENT(PARAMETER)s.

The tradition within the GEMPACK-using community is to declare as VARIABLEs (rather than as parameters) any quantity which might conceivably vary.[62]

For example, you may have a model which includes tax rates which you do not (at present) intend to vary. You could declare them as COEFFICIENT(PARAMETER)s but it may be more useful to declare them as VARIABLEs. In the latter case, you can convey the fact that they do not vary by indicating that the VARIABLEs are exogenous and not shocked.[63] Later, if you wish to model the consequences of these tax rates changing, you do not have to change the model.

In Stylized Johansen, there are only two exogenous variables, the supplies of the primary factors labor and capital, so this issue does not arise. However, it does arise in most of the more serious models implemented via GEMPACK. For example, in ORANI-G (see section 1.4 of GPD-8), many quantities which do not change in most simulations (for example, household subsistence demands, various technology terms and various shifters) are declared as Variables rather than as Parameters.

### 3.3.7 TABLO Language - Syntax and Semantics

Full details of the syntax and semantics used in TABLO Input files are given in chapters 3 and 4 of GEMPACK Document GPD-2. The description in GPD-2 applies to all TABLO Input files - that is, to those containing just levels equations, just linearized ones and to those (such as the one in section 3.3.3 above) containing a mixture of levels and linearized equations. We will introduce more information about the syntax and semantics in sections 3.4 and 3.5 below (where we describe alternative TABLO Input files for Stylized Johansen, firstly one containing just linearized equations and secondly one containing just levels equations).

---

[62] This is different from the GAMS-using community, where the tradition is to declare more quantities as parameters [see section 2.5 of Kohlhaas and Pearson (2002)].

[63] If a variable is exogenous and not shocked in a simulation, the effect is the same as if it had been declared as a Parameter.

## *3.4 Linearized TABLO Input Files*

It is possible to make a TABLO Input file containing only linearized equations; we refer to such TABLO Input files as **linearized TABLO Input files**.[64]

We illustrate this by giving in full in section 3.4.1 below such a TABLO Input file for Stylized Johansen. This file is usually called SJLN.TAB when supplied with the GEMPACK Example files.

In comparison with the mixed TABLO Input file for Stylized Johansen in section 3.3.3 above, the main differences to note are as follows.

- The linear VARIABLEs are declared explicitly.
- The levels variables do not seem to be present. But in fact, in a linearized TABLO Input file, many of these are declared as COEFFICIENTs. Thus, in TABLO Input files, COEFFICIENTs have two functions.
    1. They can denote the (pre-simulation) values of a levels variable.
    2. They can denote parameters.
- In a linearized TABLO Input file, the requirement that an initial solution be obtainable from the data base means that the values of all COEFFICIENTs occurring in the linearized EQUATIONs must have their values defined (via READs or FORMULAs).
- It is necessary to provide **UPDATE** statements to tell how the data read from the data base changes in response to small changes in the relevant linear VARIABLEs. (It helps to think in terms of a multi-step simulation as described in section 2.13.3 above. After each step, the data base has to be updated to take into account changes in all the linear VARIABLEs over the step.) We give a more detailed discussion of UPDATE statements in section 3.4.4 below. One role of UPDATE statements is to provide the link between the linear VARIABLEs and the COEFFICIENTs (that is, levels variables).

We give the full TABLO Input file in section 3.4.1 and then discuss noteworthy features of it in section 3.4.2 below.

Advice about linearizing equations by hand can be found in section 9.2 in GPD-2.

[64] Indeed, in releases of GEMPACK prior to Release 5.0 (April 1993), these were the only kind of TABLO Input files allowed.

### 3.4.1 A Linearized TABLO Input File for Stylized Johansen

```
!------------------------------------------------------------------!
!               Linearized TABLO Input file for the                !
!                       Stylized Johansen model                    !
!                                                                  !
!           following the description in Chapter 3 of the text     !
!    "Notes and Problems in Applied General Equilibrium Economics" !
!       by P.Dixon, B.Parmenter, A.Powell and P.Wilcoxen [DPPW]    !
!               published by North-Holland 1992.                   !
!------------------------------------------------------------------!

! Text between exclamation marks is a comment.                     !
! Text between hashes (#) is labelling information.                !

!------------------------------------------------------------------!
!      Sets                                                        !
!------------------------------------------------------------------!

! Index values i=1,2 in DPPW correspond to the sectors called s1,s2.
  Index values i=3,4 in DPPW correspond to the primary factors,
  labor and capital.
  The set SECT below doubles as the set of
   commodities  and the set of industries. !

 SET SECT # Sectors # (s1-s2)  ;
 SET FAC # Factors  # (labor, capital) ;
 SET NUM_SECT # Numeraire sector - sector 1 #  (s1) ;
 SUBSET  NUM_SECT is subset of SECT ;

!------------------------------------------------------------------!
!  File                                                            !
!------------------------------------------------------------------!

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

!  In the DPPW names shown below,  : denotes subscript.            !
!  Thus, for example,   x:j indicates that j is a subscript.       !

!------------------------------------------------------------------!
!  Variables                                                       !
!    (All are percentage changes in the relevant levels quantities) !
!------------------------------------------------------------------!

VARIABLE (ORIG_LEVEL=Y) p_Y  # Total household expenditure #
    ! This is also  y  in DPPW  !;

VARIABLE (ORIG_LEVEL=1) (all,i,SECT) p_PC(i) # Price of commodities #
    ! This is  p:i  (i=1,2)  in DPPW  !;

VARIABLE (ORIG_LEVEL=1) (all,f,FAC) p_PF(f)  # Price of factors #
    ! This is  p:i  (i=3,4)  in DPPW  !;

VARIABLE (ORIG_LEVEL=DVCOM) (all,i,SECT) p_XCOM(i)
        # Total demand for (or supply of) commodities #
    ! This is  x:i  (i=1,2)  in DPPW  !;

VARIABLE (ORIG_LEVEL=DVFAC) (all,f,FAC) p_XFAC(f)
        # Total demand for (or supply of) factors #
    ! This is  x:i  (i=3,4)  in DPPW  !;

VARIABLE (ORIG_LEVEL=DVHOUS) (all,i,SECT) p_XH(i)
                # Household consumption of commodities #
    ! This is  x:i0  (i=1,2) in DPPW  !;

VARIABLE (ORIG_LEVEL=DVCOMIN) (all,i,SECT)(all,j,SECT) p_XC(i,j)
      # Intermediate commodity inputs #
    ! This is  x:ij  (i,j=1,2)  in DPPW  !;
```

```
VARIABLE (ORIG_LEVEL=DVFACIN) (all,f,FAC)(all,j,SECT) p_XF(f,j)
      # Intermediate factor inputs #
    ! This is  x:ij  (i=3,4; j=1,2)  in DPPW  !;

!-----------------------------------------------------------------!
!  Base data, updates and reads                                   !
!   (Base data is as in Table E3.3.1 of DPPW)                     !
!-----------------------------------------------------------------!

COEFFICIENT   (all,i,SECT)(all,j,SECT) DVCOMIN(i,j)
      ! Dollar value of inputs of commodity i to industry j ! ;

UPDATE (all,i,SECT)(all,j,SECT)
       DVCOMIN(i,j) = p_PC(i)*p_XC(i,j) ;

COEFFICIENT    (all,f,FAC)(all,j,SECT) DVFACIN(f,j)
      ! Dollar value of inputs of factor f to industry j ! ;

UPDATE (all,f,FAC)(all,j,SECT)
       DVFACIN(f,j) = p_PF(f)*p_XF(f,j) ;

COEFFICIENT (all,i,SECT) DVHOUS(i)
      ! Dollar value of household use of commodity i ! ;

UPDATE (all,i,SECT)
       DVHOUS(i) = p_PC(i)*p_XH(i) ;

!-----------------------------------------------------------------!
!  Reads from the data base                                       !
!-----------------------------------------------------------------!

READ DVCOMIN FROM FILE iodata HEADER "CINP" ;
READ DVFACIN FROM FILE iodata HEADER "FINP" ;
READ DVHOUS  FROM FILE iodata HEADER "HCON" ;

!-----------------------------------------------------------------!
!  Other coefficients and formulas for them                       !
!-----------------------------------------------------------------!

COEFFICIENT (all,i,SECT) DVCOM(i)
     ! Dollar value of total demand for commodity i ! ;

FORMULA   (all,i,SECT)
     DVCOM(i) = SUM(j,SECT, DVCOMIN(i,j)) + DVHOUS(i) ;

COEFFICIENT (all,f,FAC) DVFAC(f)
     ! Dollar value of total demand for factor f ! ;

FORMULA   (all,f,FAC)
     DVFAC(f) =  SUM(j,SECT,DVFACIN(f,j)) ;

COEFFICIENT(PARAMETER)   (all,i,SECT)(all,j,SECT) ALPHACOM(i,j)
      ! alpha(i,j) - commodity parameter in Cobb-Douglas
        production function.  It is equal to the initial share
            of commodity i in total inputs to industry j !
     ! This is  alpha:ij  (i=1,2; j=1,2) in (E3.1.4) of DPPW !  ;

FORMULA(INITIAL)  (all,i,SECT)(all,j,SECT)
                   ALPHACOM(i,j) = DVCOMIN(i,j)/DVCOM(j) ;

COEFFICIENT(PARAMETER)   (all,f,FAC)(all,j,SECT) ALPHAFAC(f,j)
      ! alpha(f,j) - factor parameter in Cobb-Douglas
        production function.  It is equal to the initial share
            of factor f in total inputs to industry j !
     ! This is  alpha:ij  (i=3,4; j=1,2) in (E3.1.4) of DPPW !  ;

FORMULA(INITIAL)  (all,f,FAC)(all,j,SECT)
                   ALPHAFAC(f,j) = DVFACIN(f,j)/DVCOM(j) ;
```

```
COEFFICIENT   (all,i,SECT)(all,j,SECT) BCOM(i,j)
      ! beta(i,j) - share of industry j in total
                                demand for commodity i !
     ! This is  beta:ij  (i=1,2; j=1,2) in (E3.2.4) of DPPW !  ;

FORMULA   (all,i,SECT)(all,j,SECT)
         BCOM(i,j) = DVCOMIN(i,j)/DVCOM(i) ;

COEFFICIENT   (all,i,SECT)  BHOUS(i)
      ! beta(i,0) - share of households in total
                                demand for commodity i !
     ! This is  beta:i0  (i=1,2) in (E3.2.4) of DPPW !  ;

FORMULA   (all,i,SECT) BHOUS(i) = DVHOUS(i)/DVCOM(i) ;

COEFFICIENT   (all,f,FAC)(all,j,SECT) BFAC(f,j)
      ! beta(f,j) - share of industry j in total demand for factor f !
     ! This is  beta:ij  (i=3,4; j=1,2) in (E3.2.5) of DPPW !  ;

FORMULA   (all,f,FAC)(all,j,SECT)
         BFAC(f,j) = DVFACIN(f,j)/DVFAC(f) ;

!-------------------------------------------------------------------------!
!  Equations (Linearized)                                                 !
!-------------------------------------------------------------------------!

EQUATION Consumer_demands # Household expenditure functions #
   (all,i,SECT) p_XH(i) =p_Y - p_PC(i)
    ! This is (E3.2.1) in DPPW !  ;

EQUATION  Intermediate_com
! From (E3.2.2) with i=1,2 in DPPW. The term p_PC(j) is included
   because of (E3.2.3) in DPPW. !
    # Intermediate demands for commodity i by industry j #
(all,i,SECT)(all,j,SECT) p_XC(i,j) = p_XCOM(j) - (p_PC(i) - p_PC(j)) ;

EQUATION  Factor_inputs
! From (E3.2.2) with i=3,4 in DPPW. The term p_PC(j) is included
   because of (E3.2.3) in DPPW. !
    # Factor input demand functions #
(all,f,FAC)(all,j,SECT) p_XF(f,j) = p_XCOM(j) - (p_PF(f) - p_PC(j)) ;

EQUATION Price_formation # Unit cost index for industry j #
    (all,j,SECT) p_PC(j) =
   SUM(t,SECT,ALPHACOM(t,j)*p_PC(t)) + SUM(f,FAC,ALPHAFAC(f,j)*p_PF(f))
     ! This is (E3.2.3) in DPPW !  ;

EQUATION Com_clear # Commodity market clearing #
    (all,i,SECT) p_XCOM(i) =
       BHOUS(i)*p_XH(i) +  SUM(j,SECT,BCOM(i,j)*p_XC(i,j))
     ! This is (E3.2.4) in DPPW !  ;

EQUATION Factor_use # Aggregate primary factor usage #
    (all,f,FAC) p_XFAC(f) = SUM(j,SECT,BFAC(f,j)*p_XF(f,j))
      ! This is (E3.2.5) in DPPW !  ;

EQUATION NUMERAIRE
               # Numeraire for the model is price of commodity 1 #
    (all,i,NUM_SECT)  p_PC(i) = 0
     ! Alternatively, this could be written as
   p_PC( "s1" ) = 0 !
    ! This is (E3.2.6) in DPPW !  ;
```

```
!-------------------------------------------------------------------!
!      Balance check for data base                                  !
!-------------------------------------------------------------------!

! In a balanced data base, total demand for commodity i, DVCOM(i)
     should equal DVCOST(i), the total cost of inputs to industry i !

! To check that total demand = total costs to industry i, remove the
   strong comment markers ![[! ... !]]! around the following lines !

![[! COEFFICIENT (all,i,SECT) DVCOSTS(i)
    ! Total cost of inputs to industry i! ;

FORMULA   (all,i,SECT)
      DVCOSTS(i) = SUM(u,SECT,DVCOMIN(u,i)) + SUM(f,FAC,DVFACIN(f,i)) ;
    ! Check that the values  of DVCOM and DVCOSTS are equal !

DISPLAY DVCOM ;
DISPLAY DVCOSTS ; !]]!

!---------end of TABLO Input file--------------------------------!
```

### 3.4.2 Noteworthy Features in the Linearized TABLO Input File

**1. DEFAULT statements**

Notice that there are no DEFAULT statements at the start of the linearized file in section 3.4.1. This is because of the convention that all TABLO Input files are assumed to begin with defaults appropriate for linearized TABLO Input files,[65] namely as if there were the following statements at the start.

```
VARIABLE (DEFAULT = LINEAR) ;
EQUATION (DEFAULT = LINEAR) ;
VARIABLE (DEFAULT = PERCENT_CHANGE) ;
COEFFICIENT (DEFAULT = NON_PARAMETER) ;
FORMULA (DEFAULT = ALWAYS) ; .
```

The purpose of the last of these is discussed under the heading "FORMULAs" below.

**2. VARIABLEs**

The linear variables are declared explicitly. We have chosen to use the same names as are declared implicitly in the mixed TABLO Input file in section 3.3.3 above. (This makes results from the 2 files easier to compare.) But we could have chosen different names.

The **(ORIG_LEVEL=…)** qualifiers may surprise you. These tell the software what to take as the pre-simulation values for the various levels variables. For example,

```
VARIABLE  (ORIG_LEVEL=Y)  p_Y  # Total nominal household expenditure # ;
```

indicates that the pre-simulation levels value of the variable p_Y is Y. Without this ORIG_LEVEL qualifier, you would not see the pre-simulation, post-simulation and changes results in Tables 2.7a or 2.7b above when you run a simulation. Similarly

```
VARIABLE  (ORIG_LEVEL=1)  (all,i,SECT)  p_PC(i) # Price of commodities # ;
```

tells the software that it can take 1 as the pre-simulation levels values of the PC(i) for each commodity i. Starting with these prices equal to one explains why it is sensible to take the pre-simulation values of the supplies XCOM(i) to be equal to the pre-simulation dollar values DVCOM(i), as indicated in

```
VARIABLE  (ORIG_LEVEL=DVCOM)  (all,i,SECT)  p_XCOM(i) #...# ;
```

---

[65] These defaults were chosen so that TABLO Input files written for versions of GEMPACK before Release 5.0 would not need altering for use with Release 5.0 and subsequent releases.

[These ORIG_LEVEL qualifiers are not necessary in the mixed TABLO Input file SJ.TAB shown in section 3.3.3 above since there the linear variable p_Y is derived automatically from Y via the declaration of the levels variable Y, so the software knows the connection between Y and p_Y.]

**3. COEFFICIENTs**

Many of the levels quantities which were declared as levels variables in the mixed TABLO Input file in section 3.3.3 are declared here as COEFFICIENTs. (For example, the dollar values DVHOUS and DVCOM. The first is READ from the data base and the second has its values assigned via a FORMULA.)

It may help to think of these COEFFICIENTs as holding pre-simulation values of the levels variables. However this is not entirely accurate in a multi-step simulation as we see below in the discussion of FORMULAs and UPDATEs.

**4. FORMULAs**

Most of the FORMULAs in the linearized file are re-evaluated at each step of a multi-step simulation. This is what the qualifier **(ALWAYS)** denotes in the DEFAULT statement shown in (1.) above. After each step of a multi-step simulation, the data base is updated and all FORMULA(ALWAYS)s are re-evaluated. For example, this ensures that DVCOM is always an accurate reflection of the DVCOMIN and DVHOUS values on the currently updated data base. [A numerical example is in section 3.4.7.]

However some FORMULAs, those with qualifier **(INITIAL)**, are only evaluated on the first step of a multi-step simulation. FORMULAs giving the value of parameters (such as those for ALPHACOM and ALPHAFAC) should only be applied initially (that is, at the first step) since the value of a parameter should not be changed.

**5. UPDATEs**

The purpose of an UPDATE statement is to tell the software how a COEFFICIENT (that is, a levels variable) changes in response to the small changes in the linear VARIABLEs at each step of a multi-step simulation.

For example, consider DVHOUS(i), the dollar value of household consumption of commodity i.

a) Suppose there were an explicit linear VARIABLE, say p_DVHOUS(i), declared giving the percentage change in DVHOUS(i). (In fact there is no such VARIABLE in the linearized TABLO Input file.) Then, in response to a change in this, the new value of DVHOUS(i) should be given by

   new_DVHOUS(i) = old_DVHOUS(i)*[1 + p_DVHOUS(i)/100].

   (On any step, the old value is the value before the step and the new value is the one put on the data base updated after the step.) We would need an UPDATE statement to indicate this. The statement could be

```
UPDATE (all,i,SECT) DVHOUS(i) = p_DVHOUS(i) ;
```

b) In fact there is no linear VARIABLE declared in the linearized TABLO Input file giving the percentage change in DVHOUS(i). However there are explicit linear VARIABLEs p_PC(i) and p_XH(i) showing the percentage changes in the relevant price and quantity. If p_DVHOUS(i) were declared, there would be a linear EQUATION connecting it to p_PC(i) and p_XH(i). This EQUATION would say that

```
p_DVHOUS(i) = p_PC(i) + p_XH(i).
```

   Thus, the procedure for updating DVHOUS(i) is

new_DVHOUS(i) = old_DVHOUS(i)*[1 + {p_PC(i)+p_XH(i)}/100] .

In fact the UPDATE statement is

```
UPDATE (all,i,SECT) DVHOUS(i) = p_PC(i) * p_XH(i) ;
```

This is interpreted by TABLO as having the correct effect (see section 3.4.6 for a numerical example). At first you may be puzzled by the multiplication sign "*" here since the percentage change in DVHOUS(i) is the SUM of p_PC(i) and p_XH(i). However, this form of UPDATE is called a PRODUCT UPDATE because it is used to update a COEFFICIENT (that is, a levels variable) which is the product of 2 or more levels variables whose percentage changes are explicit linear VARIABLEs. Here, in the levels,

DVHOUS(i) = PC(i) * XH(i)

and the "*" used in a PRODUCT UPDATE is to remind you of this levels formula.[66]

**6. Levels Prices and Quantities not Needed**

Notice that no COEFFICIENTs have been declared to hold the levels values of prices or quantities. [For example, there is no COEFFICIENT XH(i) even though there is a VARIABLE p_XH(i).] This is a fairly common feature of linearized TABLO Input files. In such files,

- normally linear VARIABLEs are declared to show percentage changes (or changes) in prices and quantities, but no explicit linear VARIABLEs are declared to show percentage changes in dollar values.
- COEFFICIENTs holding levels dollar values are declared but there are not normally COEFFICIENTs holding levels prices or quantities.

**7. Names for Levels and Linearized VARIABLEs**

As you have seen above, the levels variables required in a linearized TABLO Input file appear as COEFFICIENTs while the percentage-change (or change) variables required appear as linear VARIABLEs.

It may happen that you need on the TABLO Input file a levels variable as a COEFFICIENT and its percentage change (or change) as a VARIABLE. In this case, since TABLO Input files are not case-sensitive, you cannot follow the convention of using upper case for the levels variables or COEFFICIENTs (for example, XHOUS) and the same name partly or wholly in lower case for the associated linear VARIABLEs (for example, xHOUS).

We suggest two alternative ways around this problem.

1. Use the natural name for the COEFFICIENT version and attach 'p_' (for percentage change) or 'c_' (for change) at the start for the VARIABLE. For example,

   VARIABLE p_XHOUS(i) COEFFICIENT XHOUS(i)

2. Alternatively, use the natural name for the VARIABLE version and attach '_L' (for levels) to the end for the COEFFICIENT. For example,

[66] There are other kinds of UPDATE statements called CHANGE UPDATEs. They are less commonly needed and are documented in section 3.4.4 below.

VARIABLE xHOUS(i) COEFFICIENT XHOUS_L(i)

Although TABLO Input files are not case-sensitive (meaning that xHOUS and XHOUS are treated as being the same), we find it makes linearized TABLO Input files more readable if we consistently put linear VARIABLE names in lower case, or consistently put the first letter of all linear VARIABLE names in lower case and the rest in upper case.

### 3.4.3 Analysing Simulation Results

Now that you understand about TABLO Input files, you will want to begin learning how to analyse simulation results. This involves explaining the numbers coming out of your simulations on the basis of the equations of the model (as in the TABLO Input file), and the base data used in the simulation.

You can find a detailed hands-on analysis of the 10 percent labor increase simulation with Stylized Johansen (based on the linear TABLO Input file SJLN.TAB in section 3.4.1 above) in section 6.1 of GPD-8. This uses the program AnalyseGE which runs on Windows PCs.

### 3.4.4 Writing UPDATE Statements

The purpose of an UPDATE statement is to tell how much some part of data read changes in response to changes in the model's variables in the current step of a multi-step simulation. An introductory example was given in section 3.4.2 above.

Consider a COEFFICIENT X whose value(s) are read. There are three possibilities for the UPDATE statement for X.

1. If there is a linear VARIABLE, say p_X, in the TABLO Input file which represents the percentage change in X, then use a UPDATE statement of the form

   UPDATE X = p_X ;

2. If, in the levels, X is equal to the product of two or more quantities say Q1,Q2,...,Qn for each of which there is a corresponding percentage-change VARIABLE say p_Q1,p_Q2,...,p_Qn in the TABLO Input file, then use a UPDATE statement of the form

   UPDATE X = p_Q1*p_Q2*...*p_Qn ;

   (This type of UPDATE statement is referred to as a **PRODUCT UPDATE** since it involves updating a Levels variable which is a product of other Levels quantities.)

3. Otherwise work out an expression for the **change** in X in terms of linear VARIABLEs in the TABLO Input file and use a **CHANGE UPDATE** statement of the form

   UPDATE (CHANGE) X = <expression for change in X> ;

Of these, the second case is by far the most common and probably will cover over 90% of your UPDATE statements.[67] All three UPDATE statements in the linearized TABLO Input file for Stylized

[67] Readers familiar with Release 4.2.02 of GEMPACK may have expected to see an EXPLICIT UPDATE statement of the form
UPDATE (EXPLICIT) X = X + <expression for change in X> ;
While this form of an UPDATE statement is still accepted, we recommend that you do not use statements of this kind in the future and that you change any such statements in old TABLO Input files to UPDATE(CHANGE) statements. The numerical accuracy of solutions obtained via Gragg's method or the midpoint method increases greatly if you use the UPDATE(CHANGE) form.

Johansen are of this form (see section 3.4.1 above). The first is easy to use if it applies. Of course if COEFFICIENT X has one or more arguments, the UPDATE statements also have the appropriate quantifiers, for example (all,i,SECT), in them (as in those in Stylized Johansen in section 3.4.1). Note also that only COEFFICIENTs whose values are READ or assigned via a FORMULA(INITIAL) in the TABLO Input file should be UPDATEd.

In case 3. above, the expression for the change in X is obtained by linearizing the levels equation connecting X to other levels variables whose associated linear variables have been declared in the TABLO Input file. See section 4.11.7 of GPD-2 for a worked example.

More details about Updates, including examples, can be found in section 4.11 of GPD-2.

### 3.4.5 Numerical Versions of Linearized Equations

In this section we look at numerical versions of the linearized equations in SJLN.TAB. In two subsections below, we look at the numerical consequences of Update statements in SJLB.TAB (section 3.4.6) and at how the values of the Coefficients and the numerical equations are recalculated during each step of a multi-step calculation (section 3.4.7).

In our experience, some users are keen to have detailed information about these topics while many are not very interested. Since a detailed knowledge about these topics is not essential for doing effective modelling, **you should feel free to skip these sections**. You can always refer back to them later on, if and when you need to.

Here we look at the numerical version of the linearized equation for market clearing of commodities Com_clear. The equation is

```
Equation Com_clear   (all,i,SECT)
p_XCOM(i) = BHOUS(i)*p_XH(i) + SUM(j,SECT,BCOM(i,j)*p_XC(i,j)) ;
```

There are really 2 equations here, one for each sector ("s1", "s2").
The BHOUS and BCOM Coefficients are shares. When evaluated at the base data values (see Table 2.1.1a above), they have the values

BHOUS("s1") = 2/8 = 0.25 BHOUS("s2") = 4/12 = 0.333333

BCOM("s1","s1") = 4/8 = 0.5 BCOM("s1","s2") = 2/8 = 0.25
BCOM("s2","s1") = 2/12 = 0.166667 BCOM("s2","s2") = 6/12 = 0.5

At the start of the simulation (that is, on the first Euler step – see section 2.13.3 above), the two equations are

p_XCOM("s1") = 0.25 * p_XH("s1") + 0.5 * p_XC("s1","s1") + 0.25 * p_XC("s1","s2")

p_XCOM("s2") = 0.333333* p_XH("s2") + 0.1666667* p_XC("s2","s1") + 0.5 * p_XC("s2","s2")

This is why we call BHOUS and BCOM coefficients since their values are what are usually called the **coefficients** in the above equations.[68] The **unknowns** p_XCOM("s1"), p_XH("s1"), p_XC("s1","s1") and p_XC("s1","s2") are the Variables in the first of these equations.

When GEMPACK solves the equations above, all the variables are put onto one side so that the equation says that some expression is equal to zero. The equations above are rewritten as

1.0*p_XCOM("s1") **– 0.25** *p_XH("s1") – 0.5 *p_XC("s1","s1") – 0.25*p_XC("s1","s2") = 0

---

[68] Consider the following two linear equations in two unknowns (or variables) $x$ and $y$:
$2x + 3y = 5$, $7x – 3y = 4$.
Here the numbers 2,3,7,-3 are the coefficients. You probably learnt to solve such systems of linear equations in high (or secondary) school.

**1.0***p_XCOM("s2") – 0.333333*p_XH("s2") – 0.1666667*p_XC("s2","s1") – 0.5 *p_XC("s2","s2") = 0

If you look at Table 2.13.1 above which represents the Equations Matrix for the linearized system, the equation Com_clear("s1") is one row of the Equations matrix. The coefficients of variable p_XH("s1") give the column for p_XH("s1") in the Equations matrix so that

- you can see that the number –0.25 goes in the Com_clear("s1") row and the p_XH("s1") column [from the second term in the first equation].
- The number 1.0 goes in the Com_clear("s2") row and the p_XCOM("s2") column [the first term in the second equation].
- and similarly for the other terms in the equations.

### 3.4.6 Numerical Examples of Update Statements

Here we consider the 4-step Euler calculation with Stylized Johansen in which the supply of labor is increased by 10 percent (and the supply of capital is fixed).

We look at the effect of the Update statements after the first step of this 4-step calculation.

During the first step, the supply of labor is only increased by 2.5 percent (one quarter of the total shock). The software solves the linear equations (those in section 3.4.5 above) to work out the consequential changes in the other quantities and prices. Some results from solving these equations are as follows:[69]

p_PC("s1") = 0 p_PC("s2") = -0.25 p_XH("s1") = 1.5 p_XH("s2") = 1.75

The Update statement for DVHOUS(i) is

```
UPDATE  (all,i,SECT)  DVHOUS(i) = p_PC(i) * p_XH(i) ;
```

which means (see point 5. in section 3.4.2 above) that

```
new_DVHOUS(i) = old_DVHOUS(i)*[1 + {p_PC(i)+p_XH(i)}/100] .
```

Hence the updated values for DVHOUS after the first step are

DVHOUS("s1") = 2*[1+{0+1.5}/100] = 2*1.015 = 2.03

DVHOUS("s2") = 4*[1+{-0.25+1.75}/100] = 4*1.015 = 4.06

Similarly the other percentage changes in prices and quantities during this first step are used to update the values of the other Coefficients DVCOMIN(i,j) and DVFACIN(f,j) which are read from the data base.[70]

---

[69] One way of checking these results for this step is to look at the Johansen results for a 1% increase in labor and multiply them by 2.5 since a shock of 2.5 percent has been applied in this first step. The Johansen results for a 1% increase in labor are given in the first column in Table 2.12.3.

[70] If you wish to display the values of some Coefficient, say DVHOUS, at all steps of a multi-step calculation, you can include the statements

```
xwrite DVHOUS to the terminal ;
dws = yes;
```

in your Command file. If you add these statements to a suitable Command file, you can check the updated values for DVHOUS after the first step. These are the values reported via the xwrite statement during the formula stage of step 2 of the calculation. [See section 6.6 of GPD-3 for information about xwrite statements and section 6.1.9 of GPD-3 for information about "dws = yes ;".]

### 3.4.7 How the Equations Are Recalculated During Each Step of a Multi-step Calculation

As we indicated in section 2.13.1 above, the values of the Coefficients may change from step to step of a multi-step calculation.

Here we look at this for the second step of the 4-step Euler calculation discussed in section 3.4.6 above.

The values of all Coefficients read from the data base are updated at the end of the first step of this calculation. During the second step, these Coefficients take these updated values.

The values taken during step 2 of all other Coefficients which are not Coefficient(Parameter)s are inferred from the relevant Formulas.[71] For example, the DVCOM(i) values during step 2 are calculated by applying the TABLO Input file Formula

```
FORMULA   (all,i,SECT)
     DVCOM(i) = SUM(j,SECT, DVCOMIN(i,j)) + DVHOUS(i) ;
```

The updated values for DVHOUS (see section 3.4.6 above) and DVCOMIN are put into the right-hand side of this Formula to give the values for DVHOUS(i) used during the second step. Similarly for all other Coefficients.

Thus, for example, the values of the BHOUS(i) are recalculated during this second step. These recalculated values are put into the relevant equations (namely the Com_clear equations).

Hence the numerical linear equations solved during step 2 may be different from those solved during step 1.

In fact, for the Stylized Johansen model, the Coefficients BHOUS, BCOM and BFAC, which look as if they may change from step to step, do not change.[72] This behaviour (which is not typical of GE models) is a consequence of the fact that all behaviour in Stylized Johansen is Cobb-Douglas.

More details about the values used and calculated during the different steps of this 4-step calculation can be found in section 6.2.2 of GPD-3.

---

[71] Note that the values of those Coefficients which are declared to be Coefficient(Parameter)s are not recalculated during step 2. These Coefficients (for example, ALPHACOM(i,j)) keep the values they were given in step 1. The associated Formulas are marked as Formula(Initial)s in the TABLO Input file to indicate that they are only applied during the first step.

[72] You can see that the values of these Coefficients do not change by adding the statements

```
xwrite BHOUS to terminal ;
xwrite BCOM to terminal ;
xwrite BFAC to terminal ;
dws = yes ;
```

to a suitable Command file. Then the values of these Coefficients at all steps of the calculation will be written to the log file. [See section 6.6 of GPD-3 for information about xwrite statements and section 6.1.9 of GPD-3 for information about "dws = yes ;".]

## *3.5 Levels TABLO Input Files*

We illustrate the construction of TABLO Input files containing only levels equations by looking at such a file for Stylized Johansen in section 3.5.1. The main difference in general from mixed TABLO Input files is in connection with behavioural equations (such as CES specifications). You should expect the levels files to contain explicit calibration FORMULAs of the kind familiar to levels modellers for calculating the values of the parameters of these functions.

A surprise with the Cobb-Douglas specification in Stylized Johansen is that, although such parameters appear in the levels equations, we do not need to calculate their values since these parameters do not appear in the linearized equations produced by TABLO. But this would not be the case if Cobb-Douglas were replaced by CES.

### 3.5.1 Levels TABLO Input File for Stylized Johansen

The main difference from the mixed TABLO Input file shown in section 3.3.3 comes from using the levels version of the behavioural equations (the first three blocks in Table 3.1.1a). These involve two parameters not present in the linearized versions of these equations, namely

| | |
|---|---|
| $ALPHA_{i0}$ | parameters in the consumer demand equations |
| $Q_j$ | parameters in the intermediate demand equations |

These are called ALPHAH(i) and Q(j) respectively in the levels TABLO Input file given later in this section. As part of the calibration phase, you would expect to have to give FORMULAs for calculating the values of these. For example, using the TABLO Input file notation,

ALPHAH(i) = PC(i)*XH(i)/Y = DVHOUS(i)/SUM(ii,SECT,DVHOUS(ii))

and it would also be possible to write down a formula for the Q(j). However, in GEMPACK, the levels equations are only used as a means of writing down the linearized equations (TABLO does this by symbolically differentiating the levels equations – see section 3.7 below). Once this has been done, the levels equations are ignored. Thus, since the linearized versions of these equations no longer involve these ALPHAH and Q parameters, it is not necessary to give FORMULAs for them.[73]

Of course, in a more complicated model, you may not be sure if similar parameters are going to appear in the linearized system. When in doubt, you can write down the TABLO Input file leaving out calibration FORMULAs for such parameters and process the file by running TABLO. If the values are needed in the linearized system, TABLO will tell you and not allow you to proceed until you have supplied calibration FORMULAs.

Another noteworthy feature of the levels TABLO file shown below is in the EQUATION E_W for the quantity called W(j) there. Variable W(j) has been introduced to simplify the "intermediate demands" and "price formation" equations. The equation E_W uses the PROD operator to express W(j) as the product of the relevant quantities.[74]

The full levels TABLO Input file is shown below.

---

[73] Indeed, if you add the FORMULA shown in the text for ALPHAH(i), TABLO will tell you that this seems to be redundant because it does not appear in the linearized system.

[74] The syntax of the PROD operator is similar to that of the SUM operator (see section 4.4.2 of GPD-2). Prior to Release 8.0, when the PROD operator was introduced, the product was converted to a SUM in SJLV.TAB by taking the logarithms of both sides. This alternative verson of equation E_W is indicated as a comment in the TABLO Input file SJLV.TAB shown below.

```
!-----------------------------------------------------------------!
!                  Levels TABLO Input File for the                !
!                     Stylized Johansen Model                     !
!                                                                 !
!      following the description in Chapter 3 of the text         !
!   "Notes and Problems in Applied General Equilibrium Economics" !
!     by P.Dixon, B.Parmenter, A.Powell and P.Wilcoxen [DPPW]     !
!             published by North-Holland 1992                     !
!-----------------------------------------------------------------!

!
  Revised February 1998 to include "GE 0" qualifiers
    to take advantage of Release 6.0 of GEMPACK which can then
    ensure that these never become negative in simulations
    carried out with user-specified accuracy.
  Revised October 2002 to use PROD operator in equation E_W.
!

!   Text between exclamation marks is a comment                   !
!   Text between hashes (#) is labelling information              !

!-----------------------------------------------------------------!
!    Set defaults for Levels model                                !
!-----------------------------------------------------------------!

EQUATION(DEFAULT=LEVELS) ;
VARIABLE(DEFAULT=LEVELS) ;
FORMULA(DEFAULT=INITIAL) ;
COEFFICIENT(DEFAULT=PARAMETER) ;

!-----------------------------------------------------------------!
!      Sets                                                       !
!-----------------------------------------------------------------!

! Index values i=1,2 in DPPW correspond to the sectors called s1,s2.
  Index values i=3,4 in DPPW correspond to the primary factors,
  labor and capital.   The set SECT below doubles as the set of
  commodities  and the set of industries. !

SET SECT # Sectors # (s1-s2)  ;

SET FAC # Factors  # (labor, capital) ;

!-----------------------------------------------------------------!
!      Levels variables                                           !
!-----------------------------------------------------------------!

! In the DPPW names shown below, : denotes subscript.             !
! For example, x:j indicates that j is a subscript.               !

Variable (GE 0)        Y    # Total nominal household expenditure #
                            ! This is also Y in DPPW ! ;

Variable (GE 0) (all,i,SECT)  PC(i)   # Price of commodity i #
                               ! This is p:i (i=1,2) in DPPW ! ;
Variable (GE 0) (all,f,FAC)   PF(f)   # Price of factor f #
                               ! This is p:i (i=3,4) in DPPW ! ;
Variable (GE 0) (all,i,SECT)  XCOM(i)
    # Total demand for (or supply of) commodity i #
                               ! This is x:i (i=1,2) in DPPW ! ;
Variable (GE 0) (all,f,FAC)   XFAC(f)
    # Total demand for (or supply of) factor f    #
                               ! This is x:i (i=3,4) in DPPW ! ;
```

```
Variable (GE 0) (all,i,SECT)  XH(i)   # Household demand for commodity i #
                               ! This is x:i0 (i=1,2) in DPPW ! ;
Variable (GE 0) (all,i,SECT) (all,j,SECT)     XC(i,j)
    # Intermediate inputs of commodity i to industry j #
    ! This is x:ij (i,j=1,2) in DPPW ! ;
Variable (GE 0) (all,f,FAC)(all,j,SECT)       XF(f,j)
    # Factor inputs to industry j #
    ! This is x:ij (i=3,4; j=1,2) in DPPW ! ;
Variable (all,j,SECT)  W(j)    #Price expression#;

!------------------------------------------------------------------!
!     Dollar values read in from database                          !
!------------------------------------------------------------------!

Variable (GE 0) (all,i,SECT)(all,j,SECT)      DVCOMIN(i,j)
    # Dollar value of inputs of commodity i to industry j # ;
Variable (GE 0) (all,f,FAC)(all,j,SECT)       DVFACIN(f,j)
    # Dollar value of factor f used in industry j # ;
Variable (GE 0) (all,i,SECT)                  DVHOUS(i)
    # Dollar value of household use of commodity i # ;

!------------------------------------------------------------------!
!     Parameters                                                   !
!------------------------------------------------------------------!

COEFFICIENT (all,i,SECT)   ALPHAH(i)     #Household parameter#;
COEFFICIENT
 (all,i,SECT) (all,j,SECT) ALPHACOM(i,j) #Commodity parameter#;
COEFFICIENT
 (all,f,FAC) (all,j,SECT)  ALPHAFAC(f,j) #Factor parameter#;
COEFFICIENT (all,j,SECT)   Q(j)          #Scale parameter#;


!------------------------------------------------------------------!
!      File                                                        !
!------------------------------------------------------------------!

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

!------------------------------------------------------------------!
!      Reads from the data base                                    !
!------------------------------------------------------------------!

READ DVCOMIN from FILE iodata HEADER "CINP" ;
READ DVFACIN from FILE iodata HEADER "FINP" ;
READ DVHOUS  from FILE iodata HEADER "HCON" ;


!------------------------------------------------------------------!
!     Formulas to calculate the Initial solution                   !
!------------------------------------------------------------------!

! FORMULAs for Y, ALPHAH(i) and Q(j) are only needed if require
   change differentiation or add the Newton correction terms. !

!   1. Formulas for initial prices   !
!..................................!

FORMULA (all,i,SECT) PC(i) = 1 ;
FORMULA (all,f,FAC)  PF(f) = 1 ;
FORMULA (all,j,SECT) W(j) = 1 ;
FORMULA (all,j,SECT) Q(j) = 1 ;
```

```
!   2. Formulas which are also equations   !
!.......................................!

FORMULA & EQUATION Comin
# Intermediate input of commodity i in industry j #
 (all,i,SECT)(all,j,SECT)
 XC(i,j) = DVCOMIN(i,j) / PC(i) ;
! Quantity   = Dollar value / price !

FORMULA & EQUATION Facin # Factor input f in industry j #
 (all,f,FAC)(all,j,SECT)
 XF(f,j) = DVFACIN(f,j)/PF(f) ;

FORMULA & EQUATION House # Household demand for Commodity i #
 (all,i,SECT)
 XH(i) = DVHOUS(i)/PC(i) ;

FORMULA & EQUATION Com_clear #Commodity market clearing #
! (E3.1.6) in DPPW !
(all,i,SECT)   XCOM(i) = XH(i) + SUM(j,SECT, XC(i,j)) ;

FORMULA & EQUATION Factor_use # Aggregate primary factor usage #
! (E3.1.7) in DPPW !
(all,f,FAC)   XFAC(f) = SUM(j,SECT, XF(f,j)) ;

!  3. Formula for initial value of Y !
!....................................!
FORMULA Y = SUM(i,SECT,PC(i)*XH(i)) ;

!  4. Formulas for the parameters   !
!..................................!

FORMULA (all,i,SECT)(all,j,SECT)
 ALPHACOM(i,j) = XC(i,j)/XCOM(j) ;

FORMULA (all,f,FAC)(all,j,SECT)
 ALPHAFAC(f,j) = XF(f,j)/XCOM(j) ;

FORMULA (all,i,SECT)
 ALPHAH(i) = PC(i)*XH(i)/Y ;

!-------------------------------------------------------------!
!  Levels Equations    (Numbers refer to DPPW)                !
!-------------------------------------------------------------!

EQUATION Consumer_demands #Household expenditure functions #
! (E3.1.9) in DPPW !
 (all,i,SECT)  XH(i) = ALPHAH(i)*Y/PC(i)  ;

EQUATION Intermediate_com
 #Intermediate demand for commodity i by industry j #
! (E3.1.10) in DPPW !
 (all,i,SECT) (all,j,SECT)
 XC(i,j) = ALPHACOM(i,j)*Q(j)*XCOM(j)*W(j)/PC(i) ;

EQUATION E_W # Define W(j) to simplify other equations #
 (all,j,SECT)
 W(j) = PROD(t,SECT,PC(t)^ALPHACOM(t,j)) *
                      PROD(u,FAC,PF(u)^ALPHAFAC(u,j)) ;
! Prior to Release 8.0, when the PROD operator was introduced,
   this equation was written as:
 LOGE(W(j)) = SUM(t,SECT,ALPHACOM(t,j)*LOGE(PC(t))) +
                      SUM(u,FAC,ALPHAFAC(u,j)*LOGE(PF(u))) ;!
```

```
EQUATION Factor_inputs  # Factor input demand functions #
                                        !(E3.1.10) in DPPW !
 (all,f,FAC) (all,j,SECT)
 XF(f,j) = ALPHAFAC(f,j)*Q(j)*XCOM(j)*W(j)/PF(f) ;

EQUATION Price_formation # Unit cost index for industry j #
                                        ! (E3.1.12) in DPPW !
 (all,j,SECT)   PC(j) = Q(j)*W(j) ;

EQUATION Numeraire
 #Numeraire for the model is price of commodity 1 (E3.1.23)#
 PC("s1") = 1 ;

!-------------end of TABLO Input file-------------------------!
```

## 3.6 Mixed, Linearized or Levels TABLO Input Files?

You are probably asking yourself which it is best to build - mixed, linearized or levels TABLO Input files. In our view, there is no single answer to this question; rather, the answer depends on your background and priorities.

We believe that, for many models, mixed representations are easiest to understand (and perhaps to build); this is why we introduced the mixed version of Stylized Johansen first. Modellers who come from a background of levels or linearized models may find these sorts of TABLO Input files easier to work with.

Perhaps the main thing to keep in mind is that it doesn't matter really which you build since (provided they are done correctly) they all produce the same results. From this point of view, the answer is to use whichever form comes most naturally to you.

A fuller discussion of this whole question can be found in Harrison *et al* (1994). In that document we expressed some concerns that mixed implementations may prove to be less efficient (that is, require more memory, or take longer to solve) than hand-crafted linearized representations. In the intervening years, PCs have become significantly faster and memory has become significantly less expensive, and we no longer think that this is a serious issue.

Advice about linearizing equations by hand can be found in section 9.2 of GPD-2.

## 3.7 TABLO Linearizes Levels Equations Automatically

When the program TABLO processes a TABLO Input file containing levels EQUATIONs and levels VARIABLEs, it converts the file to a linearized file; we refer to this as the **associated linearized TABLO Input file**. Although you may not see this associated linearized file (since the conversion is done internally by TABLO), you should be aware of some of its features.

The most important feature of this conversion is that, for each levels VARIABLE, say X, in your original TABLO Input file, there is an associated linear VARIABLE whose name is that of the original levels variable with "p_" added at the start.[75]

Also, for each levels VARIABLE in the original TABLO Input file, a COEFFICIENT with the same name as the levels VARIABLE is declared in the associated linearized TABLO Input file.

It is important to realise that the rest of TABLO (the last part of the CHECK and all of the CONDENSE and CODE stages) proceeds

**as if the associated linearized TABLO Input file were the actual TABLO Input file.**

This means that warnings and error messages given by TABLO may refer to statements in this associated linearized file rather than in your original TABLO Input file.[76]

Other features of this conversion are explained in section 2.2 of GPD-2.

---

[75] Actually this is not entirely accurate. If the levels VARIABLE is declared via a VARIABLE(LEVELS,CHANGE) statement (see section 3.3.5), the associated linear VARIABLE has "c_" at the start. The "p_" and "c_" can be changed – see section 2.2.2 of GPD-2.

[76] Prior to Release 8, the names of the associated linear variables had to be used when specifying the closure and shocks on Command files. Now you can use either the levels or the associated linear names – see section 5.10 of GPD-3.

## *3.8 Condensing Models*

In many cases models need to be reduced in size before it is practical to solve the linearized equations **C z = 0** (as in equation (1) of section 2.13.1 above). For example, with the ORANI model as in Dixon *et al* (1982), there are originally over a million equations, which it is impossible to solve directly even on large mainframe computers. The aim of condensation is to reduce the size (number of rows and/or columns) of this matrix C.

There are two main ways of reducing the size of a model before attempting to solve it. The first is by substituting out variables (ones that are to be endogenous) and the second is by omitting variables (ones that are to be exogenous and not shocked in a group of simulations).

We describe how you can use TABLO to do these in the subsections below.

As you will have noticed, TABLO begins by checking your TABLO Input file, reporting any errors it finds; we refer to this as the Check stage of TABLO. If there are no errors, you usually go on to the Code generation stage of TABLO. However, if you need to condense your model, you should choose to go on to Condensation (rather than Code generation) after the Check has been completed. Then go on to Code generation once you have carried out the condensation actions you desire.

Note that, during the first stage of TABLO (the **Check stage**), all levels EQUATIONs have been automatically linearized by TABLO and the resulting linearized equations only contain the associated linear variables (as described in section 3.7 above). All condensation is done in relation to these linearized equations and so involves only linear variables. However, when responding to prompts from TABLO, if you are substituting out, backsolving for, or omitting a variable which is declared as a levels variable, you can use either the levels name or the associated linear name.[77]

### 3.8.1 Substituting Out Variables

Suppose you want to substitute out (linear) variable **x** (all components of it) using the (linearized) equation

```
(ALL,i,COM)  x(i) = A6(i)*y(i) + z(i).
```

In carrying out the substitution for **x**, TABLO will replace every occurrence of a component of **x** in the other (linearized) EQUATIONs and any UPDATEs of the model by an expression of the form

```
A6(i)*y(i) + z(i).
```

For example, the equation

```
(ALL,k,COM)  B5(k)*(x(k) + y(k)) = 0
```

becomes

```
(ALL,k,COM) B5(k)*([A6(k)*y(k)+z(k)] + y(k)) = 0.
```

An equation you nominate to be used in the substitution of a variable may need to be manipulated by TABLO into the form **x = ...**. For example, in order to use it to substitute out variable **x**, TABLO rewrites the equation

```
(ALL,i,COM) z(i) + A8(i)*x(i) = A10(i)*t3(i)
```

as

```
(ALL,i,COM) x(i) = [1/A8(i)]*[A10(i)*t3(i)-z(i)].
```

[77] The use of levels variable names in this context was introduced in Release 8.0.

Of course this substitution would lead to a division by zero error if A8(i) were equal to zero for any commodity i. TABLO alerts you to this potential problem.

- If you are running interactively, you are asked to confirm that coefficient A8(i) is never zero. If you answer that A8 can be zero the substitution is not made.
- If you are running in batch mode (as defined in section 5.3.2 below), a warning message asking you to check that A8(i) cannot be zero is written to the terminal and to the Information file, and TABLO continues with the substitution.

If you proceed with the substitution and some value of A8 is indeed zero, the error will be detected when you run the TABLO-generated program.

In order to perform a substitution when running TABLO, you merely say which variable to substitute out and which equation to use; you can nominate the equation by giving its name or its number. Then TABLO automatically rewrites all the remaining EQUATIONs and UPDATEs.

If you substitute out a variable with k components, this reduces by k the number of rows and the number of columns of the matrix in the system Cz=0 of equations to be solved.

Because the variable substituted out no longer appears in any equations, it must be an endogenous variable. Also, you cannot see its values in any simulation, since it does not appear in the resulting Equations or Solution files.

### 3.8.2 Condensation Examples

A possible condensation of Stylized Johansen is given in Exercise 3.5 of DPPW, where, in the notation of our TABLO Input file in section 3.3.3 above, they suggest substituting out the variables p_XH, p_XC and p_XF using the (linear versions of) equations House, Comin and Facin respectively. Since these variables have 2,4 and 4 components respectively, this reduces the number of rows and columns in the system each by 10, resulting in a 17 x 19 system.[78]

In the examples below, we suggest that you make these substitutions with Stylized Johansen and then carry out a simulation with the resulting condensed model. The examples 1 to 3 below carry out the same simulation as in section 2.1 above (as in Command file SJLB.CMF) so that you can compare your results with those in section 2.7. The results for endogenous variables remaining in the condensed system should be the same, except for possible rounding errors.

**Example 1 - Making a Stored-input file for Condensation**

In WinGEM,

set the working directory to C:\SJ as described in section 2.4.3.

Open a TABLO Window via ***Simulation | TABLO Implement*** .

Select ***Options | Run interactively***

and give the responses in the box below.

Alternatively, run TABLO at the Unix/Command prompt.

[78] The GEMPACK mixed implementation of the uncondensed model is of size 27 x 29, as shown in Table 2.13.1 above. In DPPW, the ten equations connecting dollar values to prices and quantities - that is, the equations (E8), (E9) and (E10) in section 3.1.1 above - are not shown explicitly. This is why Table E3.5.1 in DPPW, showing the Equations Matrix for the condensed system, only has size 7 x 9.

Your responses in running TABLO could be as follows.

> [After each response (which is shown in bold type), we have given a comment, which starts with an exclamation mark '!'. When running the program, you should not type in the exclamation mark or the following comment – just type in the text shown in bold type. The third response is a carriage-return or ENTER.]

```
sif                ! Store inputs on a file
sjcond.sti         ! Name of Stored-input (STI) file
<carriage-return>  ! Default options (Begin the Check stage)
sj                 ! Name of TABLO Input file
sjcond             ! Name of Information file ('COND' for condense)
c                  ! Do Condensation
s                  ! substitute
p_xh               ! variable p_XH (case does not matter here)
house              ! using equation House
s                  ! substitute
xc                 ! variable XC – can use levels or linear name
comin              ! using equation Comin
s                  ! substitute
p_XF               ! variable p_XF
facin              ! using equation Facin
e                  ! Exit from Condensation
a                  ! Begin Automatic Code generation
pgs                ! Prepare output for GEMSIM (or wfp for TG program)
<carriage-return>  ! other default code options
sjcond             ! Name of GEMSIM Auxiliary files (or TG program name)
```

**User Input to TABLO to Condense Stylized Johansen**

Notice that we suggest using SJCOND as the name of the GEMSIM Auxiliary files (to distinguish them from the ones called SJ produced from Step 1 in sections 2.4.7 and 2.6.3 above).[79]

The option **sif** in first line of User Input is used to create a Stored-input file called SJCOND.STI that can be used to run TABLO with the same condensation later.[80] See Example 2 below for details about how to reuse this Stored-input file, and Example 3 on how to run a simulation with the Condensed SJ model. Look at SJCOND.STI in your text editor.

Note that, when responding to prompts from TABLO, if you are substituting out a variable which is declared as a levels variable, you can use either the levels name or the associated linear name. [In the responses shown above, you can see responses p_xh (the linear name) and XC (the levels name).][81]

More details about substituting out variables in general can be found in section 2.3 of GPD-2. In particular, this states the precise conditions under which a given equation can be used to substitute out a particular variable.

**Example 2 - Running TABLO from a Stored-input file**

Example 2 is an example of condensation of the Stylized Johansen model using an existing Stored-input file. You have just condensed the model in the previous example, storing the responses on the

---

[79] If you give the name SJ here, on many machines the new files SJ.GSS and SJ.GST will overwrite (that is, delete) the original ones produced in Step 1 of section 2.4.7 and 2.6.3. Similar considerations apply to our choice of SJCOND as the Information file name.

[80] See section 5.3 for more details about Stored-input files and the option **sif**.

[81] The use of levels variable names in this context was introduced in Release 8.0.

Stored-input file SJCOND.STI. This example repeats the process but you do not have to type in all the responses again.[82] Once you have made a Stored-input file condensing your model you can reuse the same file whenever you need to condense your model again. Most of the well-known models such as GTAP, ORANIG and MONASH have a standard condensation which is supplied in a Stored-input file, and TABLO is always run using this Stored-input file.

**If you are working using the Unix/command prompt**,

you can run TABLO by typing in

```
tablo -sti sjcond.sti
```

```
ltg sjcond
```  (if you need to Compile and link)

**In WinGEM**,

first check that the working directory is set as the subdirectory \SJ

by selecting in the main WinGEM menu,

***File | Change both default directories***

Check that the working directory is the subdirectory \SJ where you installed the files for Stylized Johansen. Then from the main WinGEM menu choose

***Simulation | TABLO Implement...***

to open a TABLO Window.

In doing a condensation, you need to say which variables are to be condensed out and which equations are to be used. This information has been prepared in a Stored-input file (or STI file) called SJCOND.STI in the previous example. To tell TABLO to use this file, choose **in the menu for the TABLO window**

***Options | Run from STI file***

and then ***Select*** the name of the Stored-input file. Choose the Stored-input file SJCOND.STI you created in the previous example.[83]

***Run*** TABLO. When TABLO has finished, view the Log file and the Information file to see that TABLO has completed successfully.

If you produced the TABLO-generated program SJCOND.FOR ***Go to Compile and Link*** and create the executable image SJCOND.EXE.

**Example 3 - Running a Simulation with Condensed SJ**

**If you are working in WinGEM**,

you need to make a new Command file to run the simulation with the

condensed model To do this, choose from the main WinGEM menu

***File | Edit file...***

and select the file SJLB.CMF. Choose from the editor menu

***File | Save As...***

---

[82] You can also modify it to carry out a slightly different version of the condensation. You will do this in Example 4 in section 3.8.4 below.

[83] In the GEMPACK examples there are also two similar files for condensing SJ. If you have a Source-code version of GEMPACK and want to create the TABLO-generated program, choose file SJCONDTG.STI. Alternatively choose SJCONDGS.STI, which will produce GEMSIM output.

and save the file as SJCOND.CMF.

Edit the file SJCOND.CMF to change the name of the Auxiliary files to SJCOND. The name of the Solution file will change automatically to SJCOND.SL4 because you have changed the name of the Command file to SJCOND.CMF (see section 2.5.1 of GPD-3). Save these changes and exit from the editor.

Run the program SJCOND.EXE or GEMSIM by choosing from the main WinGEM menu

***Simulation | TG program...*** (or alternatively ***Simulation | GEMSIM Solve...*** )

In the TG program case, you need to ***Select*** the TG Executable to be SJCOND.EXE.

In both cases, ***Select*** the new Command file SJCOND.CMF and ***Run*** the program.

[If there are errors during this run, edit the Command file to correct them and then rerun the simulation.]

Choose ***Go to GEMPIE*** and ***Run*** GEMPIE with the new Solution file SJCOND.SL4 to make a Print file SJCOND.PI5.

Compare the two Print files SJCOND.PI5 and the Print file from the original simulation called SJLB.PI5 by opening these files in your text editor.[84]

Alternatively you can use ViewSOL. First open one solution and then, without closing ViewSOL, open the second solution. Being able to have two or more solutions open at once is a useful feature of ViewSOL.

**If you are working using the Unix/command prompt**:

To create the Command file for this condensed model, copy the file **sjlb.cmf** to **sjcond.cmf**. Edit the file sjcond.cmf in your text editor to change the name of the Auxiliary files to **sjcond**. The name of the Solution file will change automatically to sjcond.sl4 because you have changed the name of the Command file to sjcond.cmf (see section 2.5.1 of GPD-3). Save these changes and exit from the editor.

Run the program **sjcond** or **gemsim** by typing in at the Command prompt:

```
sjcond -cmf sjcond.cmf
```

or

```
gemsim -cmf sjcond.cmf
```

When the simulation has finished, run GEMPIE (see Step 3 in section 2.6.2 above) to produce the GEMPIE Print file **sjcond.pi5**. Include all available endogenous results in sjcond.pi5. Then compare the results in sjcond.pi5 with those in sjlb.pi5 produced in section 2.6.2 above.

The results from the two simulations should be the same but not all variables are present in SJCOND.PI5. For example, the p_XH results are not in SJCOND.PI5 since the variable p_XH has been substituted out via the instructions in Stored-input file SJCOND.STI . You will learn more about variables present in SJCOND.SL4 after substitution in section 3.8.5 below.

[84] If you use editor GEMEDIT you can use Split Screen Editing to have one file open in the main window and the second file open in the View Window, and compare the two files. If you use TABmate, you can have the two files open in separate windows.

### 3.8.3 Backsolving for Variables

When you substitute out a (linear) variable, it is eliminated from all (linearized) equations in the condensed system and its values are not calculated (and so cannot be reported) when you carry out a simulation.

In principle, the values of a variable substituted out could be calculated after each step of a multi-step simulation by substituting the values of variables in the condensed system into the expression used to substitute out the variable in question. For example, if variable **x** has been substituted out using the equation

```
(ALL,i,COM)  x(i) = A6(i)*y(i) + z(i)
```

and if variables **y** and **z** remain in the condensed system, after each step of a multi-step simulation, we could calculate the values of x(i) by substituting in the known values of A6(i), y(i) and z(i) into the right-hand side of the equation above. This is known as **backsolving** for variable **x**.

When you substitute out a variable, you can indicate that you may want to backsolve for its simulation values. Follow the same procedure as described in section 3.8.1 above except that you initiate the substitution by responding 'b' (backsolve) rather than 's' (substitute); then you give the name of the variable and of the equation to use, as before. When you do this, the variable and equation in question are still eliminated from the condensed system.

However, when you carry out a simulation using GEMSIM or the relevant TABLO-generated program, you can elect (when choosing the cumulatively-retained endogenous variables – see section 8.1 of GPD-3) to have the values of the variable in question calculated (by backsolving). Backsolving, which is done at each step of a multi-step simulation (see section 6.2 of GPD-3), is carried after values of variables in the condensed system have been solved for, but before the updates of the data are done. Of course variables to be backsolved for must be endogenous. When you choose the closure and shocks, these variables are **not** available (and must not be referred to). But, when you choose the set of cumulatively-retained endogenous variables, these are present and you can say (for each simulation) which components (if any) of these variables you want retained on the Solution file. If no components of a backsolved variable are retained, the calculation is speeded up slightly since the calculations to backsolve for it are omitted at each step.

Note that variables marked for backsolving are eliminated from the condensed system and so do not appear in the Equations file. For this reason, if you carry out a Johansen simulation using SAGEM, the values of such variables **cannot** be calculated (or reported). They are only available if the simulation is carried out using GEMSIM or the appropriate TABLO-generated program.

### 3.8.4 Condensation Examples using Backsolving

The examples in this section carry out again the condensation of Stylized Johansen described in section 3.8.2 above, but this time you will backsolve for variables p_XH, p_XC and p_XF. To do this, you could proceed as in Example 1 in section 3.8.2 above but replace the relevant three **s** responses in Stored-input file SJCOND.STI by **b**. When you run GEMSIM or the TABLO-generated program, if you select all available variables to be cumulatively-retained endogenous, you would see that the values of the backsolved variables appear on the Solution file (and on the Extrapolation Accuracy file). Instead of re-running TABLO interactively (as in Example 1 in section 3.8.2 above), we suggest below in Example 4 that you do the backsolving by first modifying the Stored-input file SJCOND.STI you created in section 3.8.2, and then running TABLO taking inputs from this modified Stored-input file.

**Example 4 - Backsolving using TABLO**

Edit the file **sjcond.sti** and save it to a new name **sjback.sti**.

Then edit this new STI file (sjback.sti) to

(1) change the "s" above p_XH to a "b" so that the variable p_XH is backsolved for, instead of being substituted out.

(2) also change the "s" above XC and p_XF to "b".[85]

(3) change the name of the Information file and the Auxiliary files to SJBACK. [You do this by changing all occurrences of **sjcond** to **sjback**.]

Exit from the STI file saving changes.

You also need a new Command file SJBACK.CMF to run a simulation with this version of the model.

**If you are working in WinGEM,**

to create the file SJBACK.CMF, use ***File | Edit file...*** to edit SJCOND.CMF and then ***Save As...*** to change the name to SJBACK.CMF. In this file, change all occurrences of **sjcond** to **sjback**. Now exit from SJBACK.CMF, saving the changes.

Now repeat the steps in the previous two examples (Examples 2 and 3 in section 3.8.2) replacing SJCOND by SJBACK throughout. That is,

(a) Run ***TABLO*** using the STI file SJBACK.STI.

(b) If you have a Source-code version, ***Compile and link*** to produce SJBACK.EXE.

(c) Run the program SJBACK.EXE or GEMSIM by choosing from the main WinGEM menu

***Simulation | TG program...*** (or alternatively ***Simulation | GEMSIM Solve...*** )

and ***Select*** the new Command file SJBACK.CMF and ***Run*** the program.

Choose ***Go to GEMPIE*** and ***Run*** GEMPIE with the new Solution file SJBACK.SL4 to make a Print file SJBACK.PI5.

**If you are working at the Unix/Command prompt,**

make the changes above in the file **sjback.sti**.

To make the new Command file, copy the file sjcond.cmf to sjback.cmf.

In this file, change all occurrences of **sjcond** to **sjback**.

Run TABLO by typing in

```
tablo -sti sjback.sti
```

```
ltg sjcond
```
(if you need to Compile and link)

Run the program **sjback** or **gemsim** by typing in at the Command prompt:

```
sjback -cmf sjback.cmf
```
or

```
gemsim -cmf sjback.cmf
```

and then run **gempie** with the Solution file **sjback.sl4**.

Compare the files for SJCOND.PI5 and SJBACK.PI5 in the text editor (or ViewSOL) to see whether p_XH is present. Also note that the results for variables p_XC and p_XF are also in SJBACK.PI5.

---

[85] Note that, when responding to prompts from TABLO, if you are backsolving for a variable which is declared as a levels variable, you can use either the levels name or the associated linear name. The use of levels variable names in this context was introduced in Release 8.0.

**Example 5 – Substitution to Backsolving using option ASB**

An alternative way to convert from substitutions to backsolves is to use option **ASB** in TABLO. See GPD-2 section 2.3.4 for details. An example of using option ASB is given below.

To carry this out here, edit your condensation file **sjcond.sti**, add one extra line at the top of the file containing the letters "ASB". Change all references to **sjcond** to **sjback**. Save the file as **sjback2.sti**. Run TABLO with this new STI file, Compile and link if necessary, and run a simulation using the Command file SJBACK.CMF. This produces the same results as the previous Example 4 without replacing all **s** in SJCOND.STI by **b** (as you did in Example 4).

### 3.8.5 Results on the Solution File in Condensation Examples

**If you are working in WinGEM**,

open the files SJLB.SL4, SJCOND.SL4 and SJBACK.SL4 in View SOL. The three Contents pages in ViewSOL are shown below.

***Contents Page for SJLB.SL4***

| Variable | Size | No. | Name |
|---|---|---|---|
| Macros | 1 | 1 | |
| p_DVCOMIN | SECT*SECT | 1 | Dollar value of inputs of commodity i to industry j |
| p_DVFACIN | FAC*SECT | 1 | Dollar value of factor f used in industry j |
| p_DVHOUS | SECT | 1 | Dollar value of household use of commodity i |
| p_PC | SECT | 1 | Price of commodity i |
| p_PF | FAC | 1 | Price of factor f |
| p_XC | SECT*SECT | 1 | Intermediate inputs of commodity i to industry j |
| p_XCOM | SECT | 1 | Total demand for (or supply of) commodity i |
| p_XF | FAC*SECT | 1 | Factor inputs to industry j |
| p_XFAC | FAC | 1 | Total demand for (or supply of) factor f |
| p_XH | SECT | 1 | Household demand for commodity i |

***Contents Page for SJCOND.SL4***

| Variable | Size | No. | Name |
|---|---|---|---|
| Macros | 1 | 1 | |
| p_DVCOMIN | SECT*SECT | 1 | Dollar value of inputs of commodity i to industry j |
| p_DVFACIN | FAC*SECT | 1 | Dollar value of factor f used in industry j |
| p_DVHOUS | SECT | 1 | Dollar value of household use of commodity i |
| p_PC | SECT | 1 | Price of commodity i |
| p_PF | FAC | 1 | Price of factor f |
| p_XCOM | SECT | 1 | Total demand for (or supply of) commodity i |
| p_XFAC | FAC | 1 | Total demand for (or supply of) factor f |

***Contents Page for SJBACK.SL4***

| Variable | Size | No. | Name |
|---|---|---|---|
| Macros | 1 | 1 | |
| p_DVCOMIN | SECT*SECT | 1 | Dollar value of inputs of commodity i to industry j |
| p_DVFACIN | FAC*SECT | 1 | Dollar value of factor f used in industry j |
| p_DVHOUS | SECT | 1 | Dollar value of household use of commodity i |
| p_PC | SECT | 1 | Price of commodity i |
| p_PF | FAC | 1 | Price of factor f |
| p_XC | SECT*SECT | 1 | Intermediate inputs of commodity i to industry j |
| p_XCOM | SECT | 1 | Total demand for (or supply of) commodity i |
| p_XF | FAC*SECT | 1 | Factor inputs to industry j |
| p_XFAC | FAC | 1 | Total demand for (or supply of) factor f |
| p_XH | SECT | 1 | Household demand for commodity i |

Note that SJLB.SL4 and SJBACK.SL4 contain the same variables but the variables p_XH, p_XC and p_XF are not present on SJCOND.SL4 because they have been substituted out. Check that the values of the variables on the three Solution files are identical.

**If you are working at the Unix/Command prompt**,

look in the Print files **sjlb.pi5**, **sjcond.pi5** and **sjback.pi5** produced earlier by GEMPIE to see which endogenous variables results are available for in each case.

You should find that sjlb.pi5 and sjback.pi5 contain the same variables but the variables p_XH, p_XC and p_XF are not present on sjcond.pi5 because they have been substituted out. Check that the values of the variables on the three Solution files are identical.

### 3.8.6 Should All Substitutions be Backsolves?

When you substitute out a variable, this reduces the size of the condensed system irrespective of whether you say you want to be able to backsolve for it. The calculations to backsolve for it are only done if you choose to have at least one of its components retained on the Solution file. From these points of view there is little cost in saying you want to retain the possibility of backsolving for all variables substituted out.

However you should be aware that marking a variable for backsolving rather than straight substitution does increase the amount of memory required by GEMSIM or, if you choose to use a TABLO-generated program, the size of the program and the memory it requires. For this reason, it is best to mark for backsolving only variables you think you will need to report.

If you plan to use AnalyseGE (see section 2.6 of GPD-4) to assist in the analysis of your simulation results, you will find it useful to backsolve for any variables which appear in equations you wish to decompose (using AnalyseGE's point and click decomposition features). This is because the values of variables which have been substituted out (rather than backsolved for) are not available to AnalyseGE and hence you cannot decompose an equation containing such a variable. Thus, if you plan to use AnalyseGE, you have a strong incentive to backsolve for many (indeed, most) variables in your model.

If you have sufficient memory to make all substitutions backsolves, note that TABLO option ASB (see Example 5 in section 3.8.4) provides a simple way of achieving that.

### 3.8.7 Omitting Variables

If, in a group of simulations, all components of a (linear) variable x(i) are to be exogenous and not shocked, all values (changes or percentage changes) in the linearized equations will be zero. Hence all terms in this variable could be omitted from all the linearized equations of the model. This is the idea behind omitting variables. If you omit a variable with k components, this reduces the number of columns in the matrix C by k (but does not change the number of rows).

To omit several variables during the condensation stage of TABLO, just respond 'o' (Omit) at the appropriate stage, then, when prompted, give the names of the variables to be omitted, one per line. When you come to the end of the list of variables to omit, enter a (further) carriage-return. TABLO automatically rewrites all EQUATIONs (and UPDATEs) by omitting all occurrences of this variable.

If you are omitting a variable which is declared as a levels variable, you can use either the levels name or the associated linear name.[86]

[86] The use of levels variable names in this context was introduced in Release 8.0.

If, in another group of simulations, these omitted variables are to be shocked (or made endogenous), simply carry out a different condensation in which these are not omitted (but perhaps others are).

When you decide to omit a group of variables, we suggest that you make this omission the first condensation action when you run TABLO. This will make the rest of your condensation actions simpler and so they will run slightly more quickly than if you had left the omissions until later.[87]

### 3.8.8 Condensation Actions Can Be Put on the TABLO Input File

Condensation actions (substitutions, backsolves, omits) can be put on the TABLO Input file. For example, you could include the statement

```
Backsolve p_XH using Consumer_demands ;
```

to indicate that you want to backsolve for variable p_XH using the equation Consumer_demands. Alternatively you can use the levels variable name XH as in

```
Backsolve XH using Consumer_demands ;
```

See section 2.4 of GPD-2 for details.

[87] As part of the fine print associated with omitting variables, note that TABLO may carry out some substitutions automatically as a result of an omission. For example, suppose that there is an equation saying

(all,i,COM) x(i) = A(i)*y(i)

If you tell TABLO to omit variable 'y' then, after this omission, this equation reads x(i) = 0 for all commodities; in this case TABLO recognises that all components of variable 'x' must also be zero and TABLO automatically makes this substitution in the rest of the equations (without waiting for you to say whether you want this done). In such cases, the Information file makes it clear that this substitution has been carried out.

## *3.9 Creating the TABLO Input File and Command Files for Your Own Model*

When you want to build your own model, you will usually construct the TABLO Input file by modifying one from an existing model. For example, you may wish to add some equations to an existing model. Advice for doing this can be found in section 1.3 of GPD-2.

Alternatively, you can create a TABLO Input file for your model from scratch. Suggestions about this can be found in section 1.4 of GPD-2.

Whenever you are building or modifying a TABLO Input file, you will probably want to use the Windows program TABmate (see section 2.4 of GPD-4) if you are working on a PC. TABmate assists you to identify and remove syntax or semantic errors from your TABLO Input file, as the examples in section 3.9.1 below show.

You will also ned to write Command files for simulations. In section 3.9.2 below, we show you how you can identify and correct errors in Command files.

### 3.9.1 Correcting Errors in TABLO Input Files

In the example below, we show you how to fix all errors in the TABLO Input file **sjerror.tab** which is supplied with the GEMPACK Examples.

**If you are working in WinGEM**, the Windows program TABmate can be a great help in finding errors.

> The example below shows you how to use TABmate to correct a TABLO Input file SJERROR.TAB which contains some typical errors.
>
> Check how your WinGEM is configured by selecting
>
> ***Options | Editor for TABLO Check Errors***
>
> and then slide your mouse across to click on ***Use TABmate*** .[88]
>
> Set your working directory to the subdirectory \SJ as described in section 2.4.3.
>
> Now open a TABLO window via ***Simulation | TABLO Implement...*** and then, in this window, ***Select*** the TABLO Input file SJERROR.TAB. Click on ***Run*** to run TABLO. This run will find errors and so you should see a new window titled **Error running TABLO**. In this window, click on ***Edit TABLO file***.
>
> This will put you into the Windows program **TABmate** which will open with this TABLO Input file SJERROR.TAB. Indeed, TABmate will show you the first error, which occurs at the beginning of the declaration of variable XCOM. You should see a wriggly red line under the word VARIABLE at the start of this line (line number 55 of the file). To see the reason for this error, click on the word VARIABLE which is underlined in red. You will see the reason
>
> **Expected ;**
>
> shown (also in red) in the Error status panel in the bottom right-hand half of the TABmate's bottom panel. [After a few seconds the reason will go away, but you can get it back by clicking on the red-underlined word.]

[88] This assumes that you are running version 2.1 (February 2000) or later of WinGEM. [You can see this information by selecting ***Help | About..*** from WinGEM's main menu.] WinGEM provides different ways of proceeding if a syntax or semantic error is found by TABLO during the Check stage. The default (which we describe in this section) is to use TABmate.

You can see that a semi-colon is missing from the end of the previous line (the end of the declaration of variable PF). To remedy this error, insert a semi-colon at the end of that line. TABmate does not immediately realise that you have fixed this error. However you can ask TABmate to check the file by clicking on the **TABLO Check** button near the middle of the top part of TABmate. When you click on this Check button, TABmate first saves the TAB file and then runs TABLO to check the file.

This time it gets past the previous error but finds another error, underlining the word **FACT** in red and giving **Unknown set** as the reason for this error. A moment's reflection will tell you that the name of this set is just **FAC** (not FACT), so correct this error by removing the final "T". Then click on **TABLO Check** button again. This time TABmate tells you **No error found** (in "go-ahead" green rather than "stop" red).

Now that you have removed all errors, you can return to WinGEM to continue. To do this, close TABmate (for example, by selecting ***File | Exit*** from the main TABmate menu). You will see WinGEM's **Error running TABLO** window. In this window, click on **Rerun**. Then WinGEM will rerun TABLO. This time there should be no errors and TABLO will produce either a TABLO-generated program or else output for GEMSIM as usual.

This illustrates the procedure for removing errors from TABLO Input files.

(i) Run TABLO.

(ii) Use TABmate (and its TABLO Check button) to remove all errors, then close TABmate.

(iii) Click on the Rerun button to rerun TABLO under WinGEM to produce a TABLO-generated program or output for GEMSIM.

**If you are working in Unix/Command prompt mode**,

run TABLO by typing in

```
tablo
```

Give the responses

```
<carriage-return>      ! Accept default options
sjerror                ! Name of TABLO Input file
<carriage-return>      ! Default name of Information file (sjerror.inf)
```

Then TABLO will check the file. It will report 1 syntax error and 10 semantic errors.

To identify the errors, open the Information file **sjerror.inf** in your text editor.

Search for **%%** (two % signs with no space between them). At the first occurrence you should see something like:

**Excerpt from Information file sjerror.inf**

```
     51   VARIABLE (all,i,SECT)  PC(i)   # Price of commodity i #
     52                                   ! This is p:i (i=1,2) in DPPW ! ;
     53   VARIABLE (all,f,FAC)   PF(f)   # Price of factor f #
     54                                   ! This is p:i (i=3,4) in DPPW !
     55   VARIABLE (all,i,SECT)  XCOM(i)
          ?
 %% Syntax error.
 Expected ;.

     56       # Total demand for (or supply of) commodity i #
     57                                   ! This is x:i (i=1,2) in DPPW ! ;
```

Note the **?** which points to the first letter of VARIABLE in the declaration of XCOM. The reason "Expected ;." is shown. You can see that a semi-colon has been left out at the end of the previous declaration, namely the declaration of VARIABLE PF. To fix the

error, you must add a semi-colon at the end of this statement in **sjerror.tab**. [There is no point in making any changes to the Information file sjerror.inf.] For the moment, continue looking in sjerror.inf. You can make the change to sjerror.tab a little later.

Search again in sjerror.inf for **%%**. The next error shows something like:

**Excerpt from Information file sjerror.inf**

```
    87   COEFFICIENT (all,f,FACT)(all,j,SECT)   ALPHAFAC(f,j)
                              ?
%% Semantic problem.
Unknown set.

    88    # Share of factor input f in costs of industry j # ;
```

Here the **?** is pointing to the name **FACT**. The reason is "Unknown set". A moment's reflection will tell you that the name of this set is just **FAC** (not FACT). Again this needs to be corrected in sjerror.tab.

Search again in sjerror.inf for **%%**. The next error shows something like:

**Excerpt from Information file sjerror.inf**

```
   109   FORMULA (all,i,FAC) PF(i) = 1.0 ;
                             ?
%% Semantic problem.
Unknown coefficient or variable.
```

The **?** points to "PF" and the reason is "Unknown coefficient or variable". PF is unknown because of the first error above (where the semi-colon being omitted means that TABLO did not understand the statement declaring variable PF). We call this a **consequential error** since it is only an error because of an earlier error.

It turns out that all the other errors are consequential errors.

So exit from **sjerror.inf** and edit the TABLO Input file **sjerror.tab** instead. Fix the two problems identified and save the changes.

Then run TABLO again. This time there should be no errors.

This illustrates the procedure for removing errors from TABLO Input files.

(i) Run TABLO.

(ii) Identify one or more errors in the Information file.

(iii) Fix this error or errors in the TABLO Input file.

(iv) Rerun TABLO. Repeat the steps above if there are more errors.

### 3.9.2 Correcting Errors in Command Files

GEMSIM or the TABLO-generated program processes the Command file very early, checking that the statements are as expected. We refer to errors identified at this stage as **syntax errors** in the Command file. If you have a syntax error in the Command file (for example, do not spell one of the keywords correctly), the program stops with an error as soon as the whole Command file is processed in this way. When you have a syntax error in your Command file, the error will be marked in the Log file by **%%** to indicate where the error occurs. If you look in the Log file from the simulation, search for **%%** to find the error and the message indicating what the error is. Example 1 below is an example of a syntax error.

If there are no syntax errors, the program begins the simulation. Other errors in the Command file can be indicated later during the simulation. For example, you may not have a valid closure, or you may read shocks from a text file which does not have the expected form. In these cases, the error message may not refer explicitly to the Command file. Look at the Log file to identify the error. The error is usually indicated near the end of the Log file and is usually (but not always) marked with **%%.** You will need to read the error message and interpret it. Example 2 below is an example of this kind.

**Example 1 – Syntax Error**

Run GEMSIM or the TABLO-generated program for Stylized Johansen and take inputs from the Command file **sjlberr1.cmf** (which is supplied with the GEMPACK Examples).

The run should end with an error. To find the error, edit the Log file in your text editor and search for **%%**.[89] You should see something like the following.

**Excerpt from Log file**

```
 ! Solution method information
 ! Closure

 exogenous p_xfac ;
 rest endogenous ;

 ! Solution method information

 method = euler ;
 ! (Syntax error in next line)
 stps = 1 2 4 ;
  %% Unknown keyword 'stps'

 ! Simulation part

 ! Name of Solution file is inferred from name of Command file.
 ! (See section 2.5 in the Release 7.0 version of GPD-3.)

 shock p_xfac("labor") = 10 ;

 verbal description =
 Stylized Johansen model. Standard data and closure.
 10 per cent increase in amount of labor.
   (Capital remains unchanged.);

 ! Options
 extrapolation accuracy file = yes ;
 log file = yes ;

 ! End of Command file
  (Finished reading the command file.)

  There is at least one error in your Command file.

 (To see the error(s), look at the LOG file 'gpx60.log'.)
 (Search for %% in this LOG file.)

   (ERROR RETURN FROM ROUTINE: TGRCMF)
   (E-Error in command file input)
   (ERROR RETURN FROM ROUTINE: GEMSIM)

  (The program terminated with an error.)
```

You can see that the syntax error is the incorrect spelling of "steps".

[89] If you are running under WinGEM, WinGEM will tell you the name of the Log file. If you are running at the Unix/Command prompt, the name of the Log file will be shown on the screen near the end of the screen output for this run.

To fix the problem, edit the Command file to fix this error and rerun the simulation.

If there are several syntax errors in the Command file, they will all be marked.

**Example 2 – Error Discovered Later in the Run**

Run GEMSIM or the TABLO-generated program for Stylized Johansen and take inputs from the Command file **sjlberr2.cmf** (which is supplied with the GEMPACK Examples).

The run should end with an error. To find the error, edit the Log file in your text editor and search for **%%**.[90] You should see something like the following.

**Excerpt from Log file**

```
 ALLMCO1 called with N1=4

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

 CHOICE OF ECONOMIC ENVIRONMENT

 (All components of 'p_XFAC' chosen to be exogenous.)
%% Not all variables have been specified exogenous or endogenous.

  (ERROR RETURN FROM ROUTINE: ENINCF)
  (E-not all variables specified exogenous or endogenous)
  (ERROR RETURN FROM ROUTINE: ENINI )
  (ERROR RETURN FROM ROUTINE: TGEN  )
  (ERROR RETURN FROM ROUTINE: GEMSIM)
(Incomplete new BCV file has been deleted.)

Inputs have been taken from the Command file
C:\SJ\sjlberr2.cmf

 (The program terminated with an error.)
```

We have indicated the error in bold type. [The error in the example above is because the statement "rest endogenous ;" has been commented out. To fix it remove the exclamation mark at the start of the line.]

In general, once you have identified the source of the error, edit the Command file to fix this error and rerun the simulation.

Following the error, there is a trace-back string of subroutines. This trace-back string is probably of no use to you but can be helpful to the GEMPACK developers when tracing bugs in the GEMPACK code. If you need help with an error, it will be helpful if you save the Log file and send it to us when you report the problem.

[90] If you are running under WinGEM, WinGEM will tell you the name of the Log file. If you are running at the Unix/Command prompt, the name of the Log file will be shown on the screen near the end of the screen output for this run.

# CHAPTER 4

# 4. Constructing Data Files

When you prepare the TABLO Input file for a model, you work out how much data is required (see section 3.2 above) and what it must represent. Then comes the (often difficult and time-consuming) task of assembling the actual data (numbers); we say nothing about this here. Once that has been done, you must create files containing these numbers which can be read by the GEMPACK programs. These files can be

(i) GEMPACK Header Array files (which are binary files) or

(ii) GEMPACK text data files.

You can create **GEMPACK text data files** using your favourite text editor or in a spreadsheet. The syntax required for text files is introduced in section 4.1.2 below, and is explained in detail in chapter 6 of GPD-4.

While text data files may be attractive because you can create or change them via an editor, they are not very practical, especially for large models. There is a danger that data on them will be assigned to the wrong COEFFICIENTs if the order of the READ statements in your TABLO Input file does not match the order of the data on the file (or if you have two or more data files for the model).

For these reasons, Header Array files are the default files recognised by TABLO, GEMSIM and TABLO-generated programs.

**Header Array files** are binary files which contain one or more arrays containing data values. An individual array of data on a Header Array file is accessed by referring to the unique 4-character identifier known as a **Header** for that array of values. See section 4.1.1 below for more details.

The GEMPACK programs MODHAR and ViewHAR make it fairly easy for you to create these files and to subsequently modify the data on them[91], while the programs SEEHAR and ViewHAR enable you to examine the data on these files.

The routine way of creating a Header Array data file is to first create one (or more) text data files which between them contain the arrays (or matrices) of data. [These text files can be created via an editor or from spreadsheet output or, occasionally, by writing a program.] Then it is easy to run MODHAR to create a Header Array file containing these arrays of data at the appropriate header (as stated in the TABLO Input file). We illustrate this for Stylized Johansen in the section 4.2 below.

Section 4.3 below contains examples which show how MODHAR can be used to modify data on Header Array files.

GEMSIM and TABLO-generated programs can be used to write Header Array or text data files. We give a hands-on example and some details in section 4.4 below.

Section 4.5 below contains hands-on examples which introduce how ViewHAR can be used to create a Header Array file or to modify data on a Header Array file. MODHAR and ViewHAR are compared in section 4.6.

---

[91] MODHAR is introduced in sections 4.2 and 4.3 below and is fully documented in chapter 3 of GPD-4. The use of ViewHAR to create Header Array files, and to modify data on them, is introduced in section 4.5 below.

When you construct a data base for a model, it is important to check that it is balanced. We give some examples in section 4.7 below.

We provide a table in section 4.8 which summarises the different programs you can use to convert data files of one type (for example, a text data file) to another type (for example, a Header Array file).

There are many techniques used in preparing and modifying data. In this chapter we only scratch the surface. We recommend sources of further information in section 4.9.

## *4.1 Header Array and Text Data Files*

### 4.1.1 Header Array Files

Header Array files should usually be preferred to text files as the means of holding data for models, especially large ones.

In a Header Array file, each array of data has an associated 4-character **header** which is used to refer to the array. Headers usually consist of letters A-Z and/or digits 0-9. Different arrays must have different headers. The case (upper or lower) of the header is not significant. (For example, you cannot have one array on a Header Array file with header 'ABCD' and another on the same file with header 'AbCd'.) Headers starting with the letters 'XX' are reserved for internal program use so an error occurs if you choose a header starting with 'XX'.

Note that, in addition to its header, each array of data has an associated **long name** (up to 70 characters long) which can contain a description of the data in the array.

Section 3.1 of GPD-4 contains a few more details about Header Array files.

Each array can have set and element labelling (which indicates, for example, the names of the commodities associated with each number) - see chapter 5 of GPD-4 for details.

Header Array files are binary files so they cannot be printed or edited directly. Because of this, GEMPACK provides a number of utility programs for accessing them. These include

| | |
|---|---|
| SEEHAR | For looking at the actual data on a Header Array file |
| SUMHAR | For summarising the contents of a Header Array file |
| ViewHAR | For looking at, and modifying, actual data on a Header Array file |
| MODHAR | For modifying the data on a Header Array file |

Details about SEEHAR and SUMHAR are given in chapter 4 of GPD-4, while MODHAR is documented in chapter 3 of GPD-4. ViewHAR (which is only available on Windows PCs) has been introduced in chapter 2 above. Further details about ViewHAR can be found in section 4.5 below and in section 2.2 of GPD-4.

### 4.1.2 Text Data Files

These are the principal means of **importing data** into GEMPACK **from elsewhere**, including non-GEMPACK programs such as spreadsheets, and of **exporting data** (for example, to report-writing software). [Details about the syntax required in GEMPACK text data files are given in chapter 6 of GPD-4.]

For example, if you prepare data for your model using a spreadsheet program, you can write it out from the spreadsheet as a text file (using the widely supported **comma-separated values** or **CSV** format). Then, using MODHAR much as in section 4.2 below, you can create a GEMPACK Header Array file containing the data. If you subsequently wish to modify the data, you can

- use SEEHAR to write it out in CSV format,
- then import it into your spreadsheet and modify the data there,
- then write out the modified data in CSV form, and
- then rerun MODHAR to create a Header Array file containing the modified data.

Note that, even if you are running GEMPACK on a Unix machine and your spreadsheet program is on a PC, you can still do this since transferring text files between PCs and Unix machines is easy (using FTP, for example).

Text data files can be used to hold the data base for models (rather than using Header Array files). They are especially suitable for small models which only require one data file. An example is the intertemporal model TREES (see section 1.8 of GPD-8); you might like to look at its text data file[92] TREES20.DAT (in the examples subdirectory of a machine on which GEMPACK is installed) and the TABLO Input file TREES.TAB to see how this data is accessed.

However using text data files for large models or models with two or more data files is not recommended. The order of the data on text files must be identical to the order of the READ statements in the TABLO Input file otherwise errors may occur without any warning. (With Header Array files, the header in the READ statement ensures that the correct data is always matched with the COEFFICIENTs.) Other problems and drawbacks with using text files as data files for models are indicated in section 4.9.1 of GPD-2. For large or complex models we recommend that you use Header Array files.

[92] Note that the file TREES20.DAT for Release 5 or later of GEMPACK is slightly different from that shown near the end of section 6 in Codsi *et al* (1992), which follows the older format required for Release 4.2.02 of GEMPACK.

## *4.2 Constructing the Header Array Data File for Stylized Johansen*

You have looked at the Header Array data file SJ.DAT for Stylized Johansen in section 2.4.4 or 2.6.1 above. In this section we take you through the steps to construct that file SJ.DAT. We recommend that you carry out these steps on your own computer.

As you have seen in section 3.2.1 above, we need three arrays of data for DVCOMIN, DVFACIN and DVHOUS; these are of size 2 x 2, 2 x 2 and 2 respectively. The matrices of data required are as shown in the data base in Table 2.1.1a of section 2.1.1 above, namely

DVCOMIN

| | |
|---|---|
| 4.0 | 2.0 |
| 2.0 | 6.0 |

DVFACIN

| | |
|---|---|
| 1.0 | 3.0 |
| 1.0 | 1.0 |

DVHOUS

| | |
|---|---|
| 2.0 | 4.0 |

**Step 1 - Create a GEMPACK Text File Containing the Data**

The first step is to create a text file containing the data. For each array of data on a text file, there are two parts:

- the **"how much data" information**, and
- the **actual data**.

The "how much data" information begins with the size of each dimension (or argument on the TABLO Input file) of the array. For example. this is just '**2**' for DVHOUS and is '**2 2**' for DVCOMIN since it is an array of size 2 x 2. Then, on text files for input to MODHAR, it is usual to include the header and long name which will be associated with the array on the Header Array file to be created, following the syntax

header "<header>" longname "<long name>"

The header must agree with that specified for the relevant READ statement on the TABLO Input file (see section 3.3.3 above).

The name of the Coefficient usually associated with this data can be indicated following the syntax

Coefficient <coefficient name followed by set arguments>

Finally, the "how much data" information must end with a semi-colon ';'. For example, the "how much data" information for DVCOMIN is

```
 2  2    header "CINP"   longname
"Intermediate inputs of commodities to industries - dollar values"
    coefficient DVCOMIN(SECT,SECT);
```

(The long name, which can describe the data in some detail, and can be up to 70 characters long, is enclosed in quotes " " and must all be on one line of the file. It cannot be split over two lines. The coefficient name, which is optional, is the coefficient in the associated TABLO Input file normally associated with this data - see section 6.1.5 of GPD-4.)

The actual data is, by default, in ROW order, with each row starting on a new line of the file.[93]

For example, for DVFACIN, the data part is

```
1.0  3.0      ! first row of the matrix of data
1.0  1.0      ! second row of the matrix of data
```

The full text data file for the Stylized Johansen model is shown below.

**Text Data File SJDAT.TXT for Stylized Johansen**

```
! Text data file (usually called sjdat.txt) which can be used
!  with MODHAR via the 'at' option to create a Header Array
!  file (usually called sj.dat) to be the file with logical
!  name 'iodata' referred to in the TABLO Input
!  file (usually called sj.tab) for the Stylized Johansen model.
! For the use of this text data file, see section 4.2 of
!  GEMPACK document GPD-1.
! The data are as set out in Table 2.1.1a of GPD-1.
!
! DVCOMIN - dollar values of commodity inputs to current production
!
2 2 header "CINP"     longname
  "Intermediate inputs of commodities to industries - dollar values"
    coefficient DVCOMIN(SECT,SECT) ;
 4.0   2.0
 2.0   6.0
!
! DVFACIN - dollar values of primary factor inputs to current production
!
2 2 header "FINP"    longname
   "Intermediate inputs of primary factors - dollar values"
     coefficient DVFACIN(FAC,SECT) ;
 1.0   3.0
 1.0   1.0
!
! DVHOUS - dollar values of household use of commodities
!
2  header "HCON"     longname
   "Household use of commodities - dollar values"
     coefficient DVHOUS(SECT) ;
 2.0   4.0
!
!  End of file
!
```

The text after the exclamation marks contains comments. Such comments, which will be ignored by the program MODHAR when it reads these files, can come anywhere in the file.[94]

---

[93] Alternatively, the data can be in column order if the word 'col_order' is in the "how much data" information. Full details of the syntax and different orders are given in chapter 6 of GPD-4.

[94] A new feature of Release 5.1 of GEMPACK was that these comments can be in the data parts of the file, when the data is in "row_order" (the default) or "col_order" (but not when it is in spreadsheet style). See chapter 6 of GPD-4 for details.

The comments, which can make the file self-documenting, follow the same syntax as for comments in terminal input or Stored-input files. That is, they start with a single exclamation mark '!' and end at the end of the line. A comment can be continued over several lines by putting an exclamation mark in the first column of the next line. [See section 5.2 for more details.]

The actual data read by MODHAR is shown in the box above in bold font.

Note that the file **SJDAT.TXT** shown above is usually supplied with GEMPACK and so should be in the relevant directory on your computer. [If not, you will have to create this text file using a text editor.] We suggest that you open this file in your text editor and find the three arrays of data and their associated "how much data" information. You will use this file in Step 2 below.

**Step 2 - Run MODHAR to Create a Header Array File**

The 3 arrays of data can be put on to a GEMPACK Header Array file by running the program MODHAR. We reproduce below the commands for running MODHAR to do this.

We encourage you to actually run MODHAR using this input to make your own copy of the data file for Stylized Johansen.

- If you are working on a Windows PC, run WinGEM. Make sure that both default directories point to the directory (usually c:\sj) in which you put the files for Stylized Johansen (see section 2.4.3 above). Click on menu item **Programs** and select item ***Run programs interactively*** and then click on **MODHAR** . The program MODHAR will begin to run in a DOS box. Give the responses shown in the box below.
- If you are working on a Unix machine, change directory into the directory in which you put the files for Stylized Johansen (see section 2.6 above). Then type
  **modhar**
  (or whatever command is necessary to start MODHAR running) and then enter the responses in the box below.

[After each response (which is shown in bold type), we have given a comment, which starts with an exclamation mark '!'. When running the program, you should not type in the exclamation mark or the following comment – just type in the text shown in bold type. The first response is a carriage-return or ENTER.]

**Input for MODHAR to Recreate Data File for Stylized Johansen**

```
<carriage-return> ! Use default program options
n                 ! Not based on old file (we are creating a new one)
sj2.dat           ! Name of file to be created
             ! (Now comes the input saying you wish to add all arrays
             !  from the text file 'sjdat.txt'.)
at                ! Add arrays from a text file
sjdat.txt         ! The name of the text file
a                 ! Add all arrays from this file
             !  (Now the end of the program)
ex                ! Exit (There is no more data to add.)
<Your name>       ! Your name[95]
Standard input-output data for the Stylized Johansen model. ! History
**end             ! end the history
y                 ! Yes, this history is what I wanted
```

[95] On some machines (not PCs) you will be asked to enter the date after this prompt about your name.

When MODHAR has finished, you may wish to check that the file SJ2.DAT has been created.[96] You can examine the data on it by running the GEMPACK program SEEHAR (or ViewHAR if you are using WinGEM).

As you can see from this, it is easy to use MODHAR to create a Header Array file with any number of arrays of data (once you have prepared the text file containing the data). It is also easy to prepare a file which contains all these inputs since the responses required are easily predicted. Such a file is called a Stored-input file (see section 4.3 below for an example).

**Step 3 - Run a Simulation with Zero Shocks to Add Set and Element Labelling**

Ideally Header Array files should include set and element labelling (see chapter 5 of GPD-4). Then, when the data on these files is examined (for example, via the programs SEEHAR or ViewHAR, whose use has been described in chapter 2 above), this labelling makes it clearer which commodities (etc) the data correspond to. The file SJ2.DAT produced in Step 2 above does not yet have this labelling.

There are various ways of adding the labelling. Perhaps the simplest is to run a simulation in which zero shocks are given. The updated data will then be identical (in values) to the original data. However, the software adds set and element labelling to this updated data.[97]

To do this for the Stylized Johansen data, you can modify the Command file SJLB.CMF to change the shock to p_XFAC("labor") to 0 (rather than 10). Also change the name of the pre-simulation data to SJ2.DAT (the file created in Step 2 above) and the solution method to Johansen. You should also change the name of the Command file to something different - say SJLABEL.CMF.[98] You should also change the name given to the updated data in that file, perhaps changing the relevant line to

```
updated data file iodata = sjlabel.dat ;
```

Then carry out a simulation (as in the Step 2 part of section 2.4.6, 2.4.7, 2.6.2 or 2.6.3 above), taking inputs from this Command file SJLABEL.CMF. You can then use the "updated" data file SJLABEL.DAT in place of SJ2.DAT since it has the same data and also set and element labelling.[99]

**Alternative to Step 3 – Write a TABLO Input file**

You can also add set and element labelling (row and column labels) to the Header Array file by writing a very simple TABLO Input file.

For example, to label the Header Array "HCON" which you want to label with the sectors (s1, s2), read in the unlabelled array and write it to the new Header Array file. The resulting array on the new data file will be labelled with the element names in the set SECT.

---

[96] We suggest that you call this output file SJ2.DAT (rather than SJ.DAT) for two reasons. Firstly so as not to overwrite the file supplied with GEMPACK. Secondly because we suggest you add set and element labelling in Step 3 below.

[97] Other ways of adding set and element labelling are described in section 5.3 of GPD-4.

[98] The file SJLABEL.CMF is usually supplied with GEMPACK.

[99] If you wish, you can use ViewHAR (as in section 2.4.4 above) or SEEHAR (as in section 2.4.17 of GPD-8) to see the difference between the unlabelled SJ2.DAT produced after Step 2 and the labelled SJLABEL.DAT produced after Step 3. [Readers who work at the Unix/Command prompt can use SEEHAR as in section 2.6.1 above to see this difference.]

```
SET SECT # Commodities # (s1, s2) ;

FILE  data # Original data – Header array file # ;
FILE (NEW)  new_data # Base data - Header Array file # ;

COEFFICIENT (all,i,SECT) DVHOUS(i)  ;


READ DVHOUS FROM FILE data   HEADER "HCON" ;
WRITE DVHOUS TO FILE new_data HEADER "HCON" ;
```

**TABLO Input File SJLAB1.TAB for Adding Set and Element Labelling**

To do this, create a TABLO Input file **SJLAB1.TAB** containing the statements above. Also create the Command file **SJLAB1.CMF** containing the lines below.

```
Auxiliary files = sjlab1 ;

! Input data files
file data = sj2.dat ;

! Output (labelled) file
file new_data = sjlab1.har ;
```

**Command file SJLAB1.CMF for Adding Set and Element Labelling**

Run TABLO to process TABLO Input file SJLAB1.TAB as in the Step 1 part of section 2.4.7 or 2.6.3 above to produce output for GEMSIM (for simplicity). Then run GEMSIM taking inputs from the Command file SJLAB1.CMF as in the Step 2 part of section 2.4.7 or 2.6.3 above. This should produce the new Header Array file **SJLAB1.HAR** containing just header "HCON" and labelled DVHOUS data.[100]

Use ViewHAR (Windows PC) or SEEHAR (Unix/Command prompt) to look at the data in SJLAB1.HAR and to check that there are set and element labels.

[100] There is nothing corresponding to the Step 3 part of section 2.4.7 or 2.6.3 above because SJLAB1.TAB is a data-manipulation TABLO Input file. (It has no EQUATIONs as you would find in a TABLO Input file for an economic model.) To add the arrays for DVCOMIN and DVFACIN to the file SJLAB1.HAR, add statements in SJLAB1.TAB declaring these Coefficients, and add READ and WRITE statements for them. Then rerun TABLO and GEMSIM.

## *4.3 Modifying Data Using MODHAR*

Once you have created the Header Array file or files for your model, you may wish to change the data. You can do this either by editing the text file used with MODHAR to create the original file (as in section 4.2 above) and then re-running MODHAR, or by running MODHAR to modify the data on the Header Array file directly. Below we show you how to carry out this second alternative.

**Example 1 – Replacing one array on an existing file**

Below we give a simple example of replacing the data at header "CINP" with a modified version of the data. The modified array is on a Header Array file called **SJNEWC.HAR** which is supplied with the GEMPACK examples and so should be in the directory in which you put the Stylized Johansen files (see section 2.4.2 or 2.6 above).

We suggest that you run MODHAR (as in Step 2 of section 4.2 above) and give the following responses.[101] The new file (SJMOD.DAT) created will have this new version of the array incorporated in it.

When running MODHAR, it is a good idea to make a Stored-input file containing the MODHAR responses using the program option **sif**, as shown below where you will create the Stored-input file **modsj.sti**. If you want to rerun MODHAR later, to carry out the same set of operations, you can use the Stored-input file instead of typing in all the responses again. See section 5.3 for more details about Stored-input (STI) files.

**Responses to MODHAR to Modify Data from Stylized Johansen**

```
sif                ! Save inputs on Stored-input file
modsj.sti          ! Name of Stored-input file
<carriage-return>  ! Use default program options
y                  ! Is based on existing file
sj.dat             ! Existing file (as supplied with GEMPACK)
sjmod.dat          ! New file to be created, containing modified data
?                  ! List commands
da                 ! Delete array
CINP               ! Name of Header to delete
<carriage-return>  ! Finish list of headers to delete
ah                 ! Add header
sjnewc.har         ! Name of file containing new array
L                  ! List headers to add
CINP               ! Header of new modified array on newdata.har
<carriage-return>  ! Finish list of headers to add
?                  ! List commands
ex                 ! Exit
a                  ! Transfer all other arrays from the old file
<your name>        ! Name of person modifying the file
Replaced header CINP with new version   ! History
from file sjnewc.har                    ! History
**end              ! End of history
```

When MODHAR finishes, use ViewHAR or SEEHAR to look at the data in the file **SJMOD.DAT** you have just created. Check that the data at header "CINP" has changed from that at this header in SJ.DAT and is the same as the data at header "CINP" in SJNEWC.HAR.

---

[101] As usual, just type the responses shown in bold, not the comments which begin with an exclamation mark '!'.

We have begun the MODHAR run above with response "sif" in order to store these responses on the Stored-input file **modsj.sti**. Suppose you find you have made a mistake in preparing the data on the file SJNEWC.HAR and need to replace the array "CINP" again at a later time. To rerun MODHAR to repeat this process of replacing the array "CINP", you can then take advantage of the fact that file modsj.sti was already created to simply type in at the command prompt

```
modhar  -sti  modsj.sti
```

(see section 5.5 below).

**Example 2 – Modifying one data item in an array**

This is an example showing how you can use MODHAR to modify data on the Header Array file produced in section 4.2 above. Suppose that you want to change the input-output data base for Stylized Johansen so that industry 2 uses an input of 7 (million) dollars' worth of commodity 2 (rather than 6 as in the standard data in section 2.1.1 above). Note that the data base will still be balanced after this single change.

(**However it hardly ever makes sense to change just one data item in a data base because often the data base is no longer balanced after the change.)**

We suggest that you run MODHAR (as in Step 2 of section 4.2 above) and give the following responses.[102] The new file (SJMOD2.DAT) created will have this change incorporated in it. Note that Stored-input file **repsj.sti** will be created in this run. You could use it to repeat the run later, if necessary.

**Example of Responses to MODHAR to Modify Data for Stylized Johansen**

```
sif                ! Save inputs on Stored-input file
repsj.sti          ! Name of Stored-input file
<carriage-return>  ! Use default program options
y                  ! Is based on existing file
sj.dat             ! Existing file (as supplied with GEMPACK)
sjmod2.dat         ! New file to be created, containing modified data
                (A typical set of responses to change data in one array)
mw                 ! Modify and write one array
CINP               ! Header whose associated data is to be modified
m                  ! Modify the data
r                  ! Replace (not scale)
o                  ! One entry
2  2  7.0          ! Replace entry in row 2 and column 2 by value 7.0
w                  ! Write the modified data
n                  ! Do not use this as basis for another array
                  (Now exit, transferring unchanged the other 2 arrays)
ex                 ! Exit
a                  ! Transfer all other arrays
<Your name>        ! Your name
Modified input of commodity 2 to industry 2       ! History
from 6 to 7.                                      ! History
**end              ! End of history
y                  ! Yes, this is what I want
```

[102] As usual, just type the responses shown in bold, not the comments which begin with an exclamation mark '!'.

When MODHAR finishes, use ViewHAR or SEEHAR to look at the data at header "CINP" in the new file **SJMOD2.DAT**. Check that the altered value of 7 is shown.

More complicated changes would require more complicated responses. But the general idea should be clear from the above. Note that, when you actually run MODHAR, it gives you information confirming initial values and new (modified) ones.

Complete documentation for MODHAR is given in chapter 3 of GPD-4.

Note also that ViewHAR can be used to modify data on a Header Array file, as we illustrate in section 4.5 below.

## *4.4 Using TABLO for Data Manipulation*

Although the main use of TABLO is to implement and solve models, the availability of WRITE statements means that it can also be used as a data manipulator. You can use TABLO to write text data files and/or Header Array files. For example, if you have a lot of data on a text file, you could use TABLO (instead of MODHAR) to create a Header Array file containing this data: read the data from the text file into suitably declared COEFFICIENTs, then write it out to the desired headers.[103] Indeed, you could also perform calculations on the data or make other changes such as rearranging the order of the arguments of some arrays of data or combining parts of an array into a single array.

Below we give a simple example of a data-manipulation TABLO Input file which does some of these things. The file should be self-explanatory. Of course, you can do much more complicated operations than those shown.

```
!    Using TABLO to manipulate data !

SET COM # Commodities # (com1 – com5) ;
SET IND # Industries #  (ind1 – ind3) ;
SET SOURCE (domestic, imported) ;

FILE (TEXT) orig_data # Original data # ;
FILE (NEW)  base_data # Base data - Header Array file # ;

! Calculate basic consumption values and put them and tax on HA file !
COEFFICIENT (all,i,COM) CONBASIC(i)  # Consumption excluding tax # ;
COEFFICIENT (all,i,COM) CONINCTAX(i) # Consumption including tax # ;
COEFFICIENT (all,i,COM) CONTAX(i)    # Tax on consumption # ;

READ CONINCTAX FROM FILE orig_data ;
READ CONTAX FROM FILE orig_data ;
FORMULA (all,i,COM) CONBASIC(i) = CONINCTAX(i) - CONTAX(i) ;
WRITE CONBASIC TO FILE base_data HEADER "CBAS"
      LONGNAME "Consumption by commodity, excluding tax" ;
WRITE CONTAX   TO FILE base_data HEADER "CTAX"
      LONGNAME "Tax on consumption, by commodity" ;

! Transfer basic government use data to HA file !
COEFFICIENT (all,i,COM) GOVBASIC(i) # Government usage # ;
READ GOVBASIC FROM FILE orig_data ;
WRITE GOVBASIC TO FILE base_data HEADER "GBAS"
      LONGNAME "Government usage by commodity" ;

! Reorder arguments of intermediate usage data !
COEFFICIENT (all,i,COM)(all,j,IND)(all,s,SOURCE) INT_ORIG(i,j,s) ;
COEFFICIENT (all,i,COM)(all,s,SOURCE)(all,j,IND) INTUSE(i,s,j) ;
READ INT_ORIG FROM FILE orig_data ;
FORMULA (all,i,COM)(all,s,SOURCE)(all,j,IND)
                        INTUSE(i,s,j) = INT_ORIG(i,j,s) ;
WRITE INTUSE TO FILE base_data HEADER "IUSE"   LONGNAME
  "Intermediate use of commodities from different sources, by industry";
```

**TABLO Input File DATA1.TAB to Carry Out Data Manipulation**

We have supplied, with the GEMPACK example files,

- the TABLO Input file **DATA1.TAB** (as shown above),

[103] You have seen a simple example SJLAB1.TAB in the Alternative to Step 3 in section 4.2 above.

- associated Command file **DATA1.CMF** (this is shown below), and
- input text data file **DATA1.DAT** (this is the file with logical name ORIG_DATA in the TABLO Input file above).

In order to carry out this (somewhat artificial) data manipulation task, we suggest that you create a new directory and copy these three files to that directory. If you are using WinGEM, set this directory as your default directory (as in section 2.4.3 above). If you are working at the Unix/Command prompt, change into this directory.

Then, in either case, proceed as follows.

- Run TABLO on DATA1.TAB to produce output for GEMSIM (for simplicity). [Follow the method in the Step 1 part of section 2.4.7 or 2.6.3 above.]
- Then run GEMSIM taking input from the Command file DATA1.CMF. [Follow the method in the Step 2 part of section 2.4.7 or 2.6.3 above.]

As you can see from the Command file DATA1.CMF (shown below), this should produce output Header Array data file DATA1.HAR. You can look at the input text data file DATA1.DAT in your text editor and look at the contents in the output Header Array file using ViewHAR (Windows PC) or SEEHAR (Unix/Command prompt).

**The Command file**

```
auxiliary files = data1 ;

! Input file
file orig_data = data1.dat ;

! Output file
file base_data = data1.har ;

! Options
log file = yes ;
```

**The Command file DATA1.CMF**

Command files for data-manipulation TABLO Input files are particularly simple. Usually they just contain statements to

- indicate the name of the TABLO Input file. This is the "auxiliary files" statement.
- indicate the name of the input data file(s) and the output data file(s).
- optionally specify that you want a LOG file via the statement "log file = yes ;" (as shown above).

You can see these parts of DATA1.CMF.

**General Points**

When you run GEMSIM or a TABLO-generated program to create a new Header Array file, the new file will always contain set and element labelling (see the Alternative to Step 3 in section 4.2 above) so you do not need to add it later.

Usually TABLO Input files for data manipulation have no EQUATIONs (and hence no VARIABLEs or UPDATEs) in them.

**Text or Header Array Input and Output files**

In the example above, reading is done from a text file and writing is to a Header Array file.

The FILE statements in DATA1.TAB indicate whether the files are text or Header Array files, and whether they are old (that is, must already exist) or new (that is, will be created) files. These FILE statements are

```
FILE (TEXT) orig_data # Original data # ;
FILE (NEW)  base_data # Base data - Header Array file # ;
```

By default, FILE statements refer to Header Array files. So the qualifier (TEXT) is necessary to indicate that the logical file orig_data corresponds to a text data file.

By default, FILE statements refer to old (that is, pre-existing) files. So the qualifier (NEW) is necessary to indicate that the logical file base_data is a new file (that is, will be created when GEMSIM or the TABLO-generated program runs).

GEMSIM and TABLO-generated programs can read from Header Array files or text data files, and can write to Header Array or text data files.

**Writing Text Data files**

If you want a text data file written, the corresponding FILE statement declaring the logical file to be written in the TABLO Input file must declare the FILE to be a NEW (this says it will be written to) TEXT file. These two pieces of information are put in as File **qualifiers** (see sections 3.5 and 4.9 of GPD-2) which appear in brackets after the word FILE in the TABLO Input file, as in

```
FILE (NEW, TEXT) ....
```

By default, text files written by GEMSIM or TABLO-generated programs have all arrays written in row order. However, you can change this to column order or spreadsheet style by adding another File qualifier (namely COL_ORDER or SPREADSHEET respectively) as in

```
FILE (TEXT, NEW, SPREADSHEET, SEPARATOR="/") ....
```

Here the qualifier SEPARATOR="/" says that you want separator '/' to appear between data items. If you specify SPREADSHEET and omit the SEPARATOR= qualifier, the separator used will be a comma, which means that the file is written in comma-separated values (CSV) format.

You could experiment with the above by modifying the FILE statement for base_data in DATA1.TAB above. To convert this output file to be a text file, use the statement

```
FILE (NEW,TEXT)  base_data # Base data - Text file # ;
```

Then the output file data1.har will be a text data file whose arrays are written in row order.

Alternatively, change the declaration to

```
FILE (NEW,TEXT,COL_ORDER) base_data # Base data - Text file # ;
```

Then the arrays in the output text file will be written in column order. The difference between row order, column order and spreadsheet order is explained in chapter 6 of GPD-4.

**More Complicated Data-Manipulation Tasks**

You can write TABLO Input files to carry out more complicated data-manipulation tasks than those in DATA1.TAB above.

TABLO can also be used to carry out aggregation of data - see Example 2 in section 4.8 of GPD-2 for a brief discussion of this. You can use ViewHAR to aggregate data in different ways (see section 2.2 of GPD-4).

## *4.5 Using ViewHAR to Create or Modify Header Array Files*

ViewHAR can be used to create Header Array file or to modify the data on a Header Array file.[104] Some hands-on examples are given below of various operations you can do in ViewHAR.

In ViewHAR there are two modes: the simplest is **Read Only mode** where you can look at a Header Array file and are not allowed to change it. To get to this mode, start ViewHAR and select from the File menu ***Use simplified, read-only menu.***

The second mode is **Editing mode** where you are allowed to modify data in the Header Array file. To get to this mode, select from the File menu ***Use advanced, editing menu.*** You must be in Editing mode to carry out the operations in sections 4.5.1 to 4.5.7 below.

In this section we show you how you can create a new file containing some arrays of data. To make these examples ones you can carry out yourself (providing, of course, you are working on a Windows PC), we give examples relating to the SETs in the TABLO Input file DATA1.TAB in section 4.4 above. These are the sets COM with elements com1 to com5, IND with elements ind1 to ind3 and the set SOURCE with elements domestic and imported.

### 4.5.1 Creating a New Header Array File

Start ViewHAR running.

From the File menu, select ***Create New File*** .

### 4.5.2 Creating a New Header

We suggest that you create an array of size 5x3 to hold domestic investment data – think of it as an array which holds the values of Coefficient INVEST_DOM(c,i) which represents the value of domestic commodity c used for investment (capital creation) in industry i.

To add this array to the file, select ***Edit | Create New Header*** .

In the Create New Header window, type in the boxes

- the 4-character header **INVD**,
- a suitable default value, say **121** (you can put in the values you want later),
- the Header type **Real [RL]**,[105]
- the number of dimensions **2** and the dimensions of the array (**5** for the first dimension and **3** for the second dimension),
- the Coeff (that is, Coefficient) name: **INVEST_DOM**,
- the Long Name: **Investment usage of domestic commodities, by industry** .

Then click on OK to create the Header. View the values in this new Header by clicking on its row in ViewHAR's Contents page. They should be all equal to the default value 121.

---

[104] However if you think you will need to repeat a process (and you probably will), we recommend that you use MODHAR (rather than ViewHAR) and save your inputs as a Stored-input file (using the **SIF** option, as introduced in section 4.3 above).

[105] String type is used for character data such as names of elements in sets.

### 4.5.3 Modifying Data At This Header

Now, we suggest that you change a couple of the values at header INVD to get used to editing numbers in ViewHAR.

- First change the value in row 1 and column 2 to 98.15. To do this, click on this number, then **right-click** (that is, click on the right-hand mouse button). You will see an edit window. Change the value to 98.15 and then click on the green tick (the left-hand button on the edit window). You will see the value change from 121 to 98.15.[106]
- Next change the entry in row 3 and column 1 to 87.1.

### 4.5.4 Creating a Second Header

Here we suggest that you create an array of size 5x2 to hold household consumption by commodity and source (domestic or imported). Think of it as an array which holds the values of Coefficient DVHOUS(c,s) which is the value of household consumption of commodity c from source s.

To add this array to the file, select ***Edit | Create New Header*** .

In the Create New Header window, type in the boxes

- the 4-character header **DOMH**,
- a suitable default value, say **243.1**,
- the Header type **Real [RL]**,
- the number of dimensions **2** and the dimensions of the array (**5** for the first dimension and **2** for the second dimension),
- the Coeff (that is, Coefficient) name: **DVHOUS**,
- the Long Name: **Household consumption of commodities** .

Then click on OK to create the Header. View the values in this new Header by clicking on its row in ViewHAR's Contents page. They should be all equal to the default value 243.1. Then change the value in row 3 and column 1 (consumption of domestic commodity 3) to 512 and the value in row 4 and column 2 (consumption of imported commodity 4) to 89.

### 4.5.5 Save the File

Select menu item **Save as…** from under ViewHAR's **File** menu. Select **All headers** and **HAR (header array) file** (the default options) in the **Choose Save Options** box. We suggest that you save this file under the name **DATA2A.HAR** and put it in the same directory as you were working in for the examples in section 4.4 above.[107]

---

[106] If you are looking at 6 decimal places in ViewHAR, you may see that the new value is 98.150002. The last figure 2 will surprise you. In common with other GEMPACK programs, ViewHAR only stores numbers in what is called single precision. This means that only the first 6 or 7 figures (here the first 7 figures are 98.15000) are stored accurately. It turns out that there is no single-precision number which shows as 98.150000. The nearest single precision number to 98.15 shows as 98.150002. You need to keep this in mind when you are checking data, as explained in section 6.8 of GPD-3.

[107] When you save this file, you will probably see the message "Incomplete labelling information was not saved for header INVD". If so, you can click on the **Help** button on this message box to get more information. The Help file explains that ViewHAR cannot save the Coefficient name (this is part of the labelling information)

Then close ViewHAR via ***File | Exit*** .

### 4.5.6 Adding Set and Element Labelling Information

ViewHAR builds up information about sets and their elements each time it opens a file.

To check this, run ViewHAR and open the Header Array file **DATA1.HAR** you created in section 4.4 above. [Make sure it is the Header Array version – you may have produced text data files with this name if you experimented with the different text outputs as we suggested near the end of section 4.4. If necessary modify DATA1.TAB to produce Header Array output and then rerun TABLO and GEMSIM to produce the Header Array version of DATA1.HAR.]

Select menu item ***Edit | View Set Library*** . You should see the sets COM (with elements com1 to com5), IND (with elements ind1 to ind3) and SOURCE (with elements domestic and imported). [To see the elements of a set, click on its name in the column headed **Set name (size)**.] There may be other sets shown as well (quite a lot if you have previously opened files for several different models in ViewHAR). Click on the **Close** button to close this Set Library window.

You can use the sets and elements in this Set Library to add set and element labelling to some or all of the arrays in any Header Array file. Below we show you how to add set and element labelling to the two arrays in file DATA2A.HAR you created in section 4.5.5 above.

Now open the file DATA2A.HAR in ViewHAR via ***File | Open Header Array File..*** . [This will close file DATA1.HAR.]

First we show you how to add set and element labelling to the array at header **INVD**. Click on the row for this array in ViewHAR's Contents screen. [It is probably in the second row of the Contents screen.] Then select menu item ***Edit | Apply/Change Set Labels*** . In the **Choose Dimension** box in the bottom left-hand corner, you should see the first dimension (**5**) is selected. We want this to correspond to the set COM of commodities. Under the **Choose Set** drop-down list on the right-hand part of the form, you probably see **No Set**. Click on arrow in this drop-down list and select the set **COM**. You should see the first dimension is now labelled COM in the **Choose Dimension** box. Click on the second dimension there (size 3). To associate this with the set IND of industries, choose set **IND** from the drop-down **Choose Set** list. Now the second dimension should be labelled IND in the **Choose Dimension** box. Click on the **OK** button. ViewHAR will tell you that it has changed the set labelling on header INVD.

Now follow the same procedure to add set and element labelling for the array at header **DOMH**. The first dimension should be labelled by COM and the second dimension by SOURCE.

Finally, change the name of this file to **DATA2B.HAR** (via the ***File | Save as…*** menu item). Then DATA2A.HAR and DATA2B.HAR contain the same data but the second file has set and element labelling on the arrays while the first does not.

You can see the difference when you look at the Contents screen. In the unlabelled case (DATA2A.HAR) the **Dimension** column shows **5*2** for header DOMH, while this column shows **COM*SOURCE** in the labelled case (DATA2B.HAR).

You can also see the difference when you look at the data in the arrays. For example, when you look at the data at header DOMH, the column labels are "1 domestic" and "2 imported" in the labelled case (DATA2B.HAR), whereas they are just "1" and "2" in the unlabelled case (DATA2A.HAR).

---

until names for the sets are supplied. [You will supply names for the sets in section 4.5.6 below.] Close the Help file and click **OK** on the message box.

### 4.5.7 Pasting from a Spreadsheet Program to ViewHAR

Open your spreadsheet program (for example, Microsoft Excel). Make a vector of 3 values. Think of these as representing 3 IND values which you wish to add to the Header Array file.

Select **Edit | Copy** to copy the values to the clipboard. [Do not select any labels or totals.]

Go back to ViewHAR. Create a new header **INDA** with a single dimension of size 3. [Follow the procedure in section 4.5.2 above. Choose the default value 0.0. Specify **No. of Dimensions** as 1.] Double-click on this header **INDA** in ViewHAR's Contents page to look at the data. To import the values from your spreadsheet program, select ***Import | Paste to Screen from Clipboard*** from ViewHAR's menu. You should see the numbers from your spreadsheet appear.

You can add set and element labelling to this array if you wish.

When you have finished with this header, select ***File | Save*** to save the file **DATA2B.HAR** with this extra header.

### 4.5.8 Other ViewHAR Capabilities

ViewHAR offers many other capabilities. You can find more information in section 2.2 of GPD-4, and complete details in the Help file supplied with ViewHAR.

## *4.6 Which to Use – MODHAR or ViewHAR?*

On a Windows PC, it is usually quicker to use ViewHAR to make a change to a data file. For a quick change this is fine. For a more elaborate change to your data base which you may wish to repeat at a later date, the ViewHAR operations involve a complicated series of actions such as selection, pointing and clicking with no detailed record of what you have just done. You need great care and good notes to repeat exactly the same set of operations.

If you think you will need to repeat a process (and you probably will), we recommend that you use MODHAR and save your inputs as a Stored-input file (using the **sif** option introduced in section 4.3 above and described in more detail in section 5.3 below). When you use MODHAR and a Stored-input file, the operations you carry out on the data file are repeatable. Running MODHAR from a Stored-input file can be incorporated in a batch file.

See also section 3.2.1 of GPD-4 for a slightly different discussion of this topic.

Further details on MODHAR are given in chapter 3 of GPD-4. The main source of information for ViewHAR is its on-line Help file.

## *4.7 Checking Data is Balanced*

When you prepare the data file(s) for a model, you must be careful to check that all the different balancing conditions are satisfied.

For example, for Stylized Johansen, costs in each industry must equal the total value of output of the corresponding commodity. In ORANI-G, there are many more balancing conditions (for example, that costs in each single-product industry equal the value of output from that industry).

It is common to write a TABLO Input file to read the data and carry out the calculations to check the balancing conditions. Such a TABLO Input file will write the results of its calculations to a file you will have to look at in order to check that values are in balance.

When you are checking balance, you need to keep in mind that GEMPACK programs only produce about 6 figures of accuracy (see section 6.8 of GPD-3).

### 4.7.1 SJCHK.TAB to Check Balance for Stylized Johansen

Supplied with GEMPACK is the file SJCHK.TAB which is used to check the balance of a data set for Stylized Johansen. If you edit this file, you will see that the values of Coefficients DVCOM (value of output of each commodity) and DVCOSTS (costs in each industry) are calculated and DISPLAYed. You will also see that a check is made to count the number of negative numbers in the data base (there should be none). We encourage you to run SJCHK.TAB on SJ.DAT to check the balance.

- If you are working on a Windows PC, run WinGEM and make sure that both default directories point to the directory in which the Stylized Johansen files are located. First run TABLO on SJCHK.TAB and produce output for GEMSIM (for simplicity). Then run GEMSIM using Command file SJCHK.CMF. When GEMSIM has run, click on ***View Input/Output files*** and look at the values of DVCOM and DVCOSTS in the Display file. Also look at the LOG file to see that there are no negatives in the data base.
- If you are working on a Unix machine, change into the directory to which you copied the Stylized Johansen files. Then run TABLO on sjchk.tab to produce output for GEMSIM (for simplicity). Then run GEMSIM taking inputs from the Command file sjchk.cmf. When GEMSIM has run, look at the values of DVCOM and DVCOSTS in the Display file sjchk.dis. Also check the LOG file sjchk.log to see that there are no negative values in the data base.

### 4.7.2 SJCHK.TAB to Check Balance of Updated Data

Whenever you carry out a simulation, the updated data should satisfy the same balance checks as the original data. [Otherwise there is something wrong with the TAB file for your model.]

It is easy to use SJCHK.TAB to check the balance of the updated data SJLB.UPD produced in the 10 percent labor increase simulation carried out via SJLB.CMF (see chapter 2). To do this, save the file SJCHK.CMF as SJLBCHK.CMF and alter the statement

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

to read

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

Then run GEMSIM taking inputs from Command file SJLBCHK.CMF. Look at the Display file SJLBCHK.DIS produced and check that SJLB.UPD is still balanced.[108]

[108] For example, the values of costs in industry 1 and of output commodity 1 are both equal to 8.4708.

### 4.7.3 GTAPVIEW for Checking and Summarising GTAP Data

The standard GTAP TAB file referred to as **GTAPVIEW** provides a summary of any GTAP data set. We have provided GTPVEW61.TAB with the GEMPACK examples. You can see an example of the use of this TAB file in section 6.8.1 of GPD-3.

The file GTPVEW61.TAB is an interesting example of a TAB file. For example, look at the set GDPEXPEND and how the values in Coefficient GDPEXP are built up via several Formulas. This shows a neat way of arranging summary data into headers on a Header Array file. You may be able to adapt some of these techniques in your own work.

### 4.7.4 Checking the Balance of ORANIG Data

See section 6.8.2 of GPD-3 for details about this.

## *4.8 Which Program to Use in File Conversion*

A summary of which programs to use in converting files from one type to another is given below in Table 4.8. It includes references to sections of this and other documents where these programs are described in more detail.

**Table 4.8: Programs to Use in File Conversion**

| Input File | Output File | Program | Option | Reference |
|---|---|---|---|---|
| Spreadsheet data | Text file (CSV) | Spreadsheet | CSV | - |
| Text file | Header Array file | MODHAR | 'at' | section 4.2 Also GPD-4 |
| or | | | | |
| Text file | Header Array file | ViewHAR | | ViewHAR help |
| Header Array file | Text file(CSV) (row order) (column order) | SEEHAR | SS<br>ROW<br>COL | GPD-4 chapter 4 |
| Text file (CSV) | Spreadsheet data | Spreadsheet | - | [Paste into spreadsheet] |
| Solution file | Header array file | SLTOHT | - | GPD-4 chapter 8 |
| Solution file | Text file(CSV) (row order) (column order) | SLTOHT | SS,SSS<br>SIR<br>SIC | GPD-4 chapter8,9 |
| Solution file | Text file(CSV) | ViewSOL | | ViewSOL help |

## 4.9 Further Information

There are many techniques used in preparing and updating data files for models. This chapter is just a very brief introduction to the topic. Suggestions for finding out more are given below.

1) Aggregation:
   ViewHAR and a program AggHAR can be used to aggregate data – see ViewHAR Help
   TABLO Input files – see example using mappings in section 4.8 of GPD-2.

2) Various operations – disaggregation, aggregation, RAS:
   these can be carried out using the DOS program DAGG written by Mark Horridge.

See the Centre of Policy Studies Web site

*http://www.monash.edu.au/policy/pgemc.htm*

for details of forthcoming courses on the preparation of data for CGE models. See the page for programs from our database expert Mark Horridge

*http://www.monash.edu.au/policy/gpmark.htm*

for useful programs such as DAGG.

# CHAPTER 5

# 5. Common Features of GEMPACK Programs

This chapter is directed mainly (but not exclusively) at users who work at the command prompt. This includes all Unix users of GEMPACK (who must work at the command prompt) and power users on Windows PCs who work at the DOS prompt from preference and/or because they use batch files to repeat tasks from time to time.

All GEMPACK main programs (apart from the Windows programs) can be run interactively at the command prompt. If you use the Windows interface WinGEM, this is not so obvious since most of the GEMPACK programs run in the background in a DOS box using Stored-input files written by WinGEM. However you can always run a program interactively[109] or from a Stored-input file. Some users prefer to run the programs in this way. [Indeed, it is the only way of running GEMPACK on Unix machines.] It has the advantage that you can make up batch jobs for tasks which you want to repeat.

Although the different GEMPACK programs are designed for carrying out different tasks, they all follow certain conventions and operate in similar ways, as explained in this chapter. For example,

- they process your responses to prompts in a consistent way (see section 5.1),
- they all allow comments starting with a single exclamation mark in input from the terminal (see section 5.2), and
- they all offer standard options (such as the ability to take all inputs from a Stored-input file and the ability to direct output to a Log file) as explained in sections 5.3 and 5.4.

It is possible to specify various files (including Command files, Stored-input and LOG files) on the command line: details are given in section 5.5. The procedure by which the programs manage memory available is described in section 5.6 while section 5.7 describes the GEMPACK error reporting scheme. Section 5.8 is about file names, file suffixes and GEMPACK file types. Section 5.9 tells you what files names are allowed on different systems.

Although sections 5.1 to 5.5 will be mainly of interest to those who run GEMPACK programs from the Command prompt (either interactively or via Stored-input files), all GEMPACK users should have some familiarity with them. The remaining sections should be of interest to all users, whether they work under WinGEM or at the command prompt.

## *5.1 Responding to Prompts*

### 5.1.1 Responding to Prompts (Upper Case or Lower Case)

When you are asked to respond to a prompt or to make a choice from a menu, the case (upper or lower) of your response usually makes no difference. For example, if you are asked to say which variable you have in mind, responses of 'p_xf' or 'p_XF' or 'P_XF' will have the same effect. Similarly, if you are asked to respond [y/n] it does not matter whether you respond "y" or "Y".

---

[109] If you use WinGEM, you can run any program interactively via WinGEM's menu item ***Programs | Run programs interactively…*** .

On most systems (but not on Unix systems), the case in which you enter file names makes no difference. However, some input to GEMPACK programs (such as the "verbal description" of a simulation) is, of necessity, case-sensitive. (In such cases the normal mix of upper and lower case letters as in a document such as this one seems the best and most readable.)

### 5.1.2 Default Response to Questions and Prompts

Many of the questions asked by the programs have a default response. Often this is indicated by one possible choice shown as a capital letter as in

Do you wish to try again? [Y/n]

where the default choice is 'Y' (meaning yes). Sometimes the default choice is mentioned explicitly in the prompt. In all cases where a default is offered (either explicitly or via a response shown as a capital letter) you can accept this default by entering a carriage-return.

### 5.1.3 How the Current Directory Affects Filename Responses

When you are working at the command prompt, the **current directory** (sometimes called the **working directory**) is the directory to which you are attached when you issue commands. You can move between directories via **cd** (change directory) commands under DOS and Unix. In many cases the full path name of your current directory is shown in each prompt, as in, for example

```
c:\gpdoc\rel80>
```

When you are working under WinGEM, the current directory is what WinGEM calls the **Default working directory**.

When you are asked for a file name, your response is interpreted **relative to the current directory**. Examples will make this clear.

In these example, suppose that your current directory is **c:\mymodels\sj** .[110]

- The simplest case is when you respond with the short name of a file. For example, if you respond
  `sj.tab`
  this means the file sj.tab in the current directory. Its full path name is
  `c:\mymodels\sj\sj.tab` [111]
- Another case is where you respond with the full path name. For example, if you respond
  `c:\oranig\og.har`
  the meaning is clear. In this case the meaning does not depend on the current directory.
- More complicated cases are where you respond with a **relative path name**. Relative path names can contain directories, and may also being with two dots which mean go one directory higher. Examples are
  `work\sjlb.cmf` which means `c:\mymodels\sj\work\sjlb.cmf` ,
  `..\temp\sjlb.cmf` which means `c:\mymodels\temp\sjlb.cmf` ,

[110] The examples we give in this section and the couple of subsections following it are written in DOS form. Unix users will easily be able to convert these examples.

[111] By the full path name we mean the path name which begins from the drive, and includes all directory or folder names from the top directory down to your current directory. By a short file name we mean the last part of a full path name (obtained by leaving out the drive and all directory parts). For example, c:\mymodels\sj\sj.tab is a full path name and sj.tab is the corresponding short name.

`\tmp\sjlb.cmf` which means `c:\tmp\sjlb.cmf` (just the "c:" is taken from the name of the current directory in this case).

When you take inputs from a Stored-input file (see section 5.3), any file names in that Stored-input file are interpreted relative to the current directory (since the Stored-input file just stores responses to prompts).

### 5.1.4 How the Current Directory Affects Filenames in Command Files

When you run a program via a Command file, all file names in the Command file are interpreted **relative to your current directory**. [They are not interpreted relative to the directory in which the Command file resides.]

For example, consider the command

```
sj  -cmf  c:\gp80\sjlb.cmf
```

Suppose that your current directory is **c:\mymodels\sj** . Then all file names in the Command file are interpreted relative to directory `c:\mymodels\sj` (not relative to directory `c:\gp80` containing the Command file). For example, the statement

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

means that the updated version of file with logical name iodata will be `c:\mymodels\sj\sjlb.upd` .

Usually WinGEM uses its "Default working directory" as the current directory (as indicated in section 5.1.3 above). However, when you run any program (GEMSIM, a TABLO-generated program or SAGEM) from a Command file under WinGEM, WinGEM sets the current directory to be the directory in which the Command file lies (even if this is different from WinGEM's default working directory).

### 5.1.5 Directories Must Already Exist

When you specify the name of a file to be created by a GEMPACK program, the directory in which the file is to be placed must already exist. GEMPACK programs do not create new directories in such cases. For example, if you respond
`c:\mysj\sjv1.out`
as the name of a file to be created, an error will occur unless directory `c:\mysj` already exists.

### 5.1.6 Windows PC GEMPACK Programs Echo Current Directory

All GEMPACK programs created on a Windows PC using the Lahey Fortran compiler LF90 or using version 5.60 or later of the Lahey Fortran compiler LF95 echo the full path name of the current directory early during their run. They also echo the full path name of the running EXE early during their run.[112]

## *5.2 Comments in Input from the Terminal*

In processing terminal input, all GEMPACK programs ignore any part of an input line starting with a single exclamation mark '!'. Since input on a Stored-input file is treated as if it were input from the terminal, this also applies to Stored-input files (which are introduced in section 5.3 below). We

[112] These features were introduced in Release 8.

recommend a liberal use of comments (beginning with a single '!') to make Stored-input files self-documenting (and also easier to modify). Since terminal input is echoed exactly (that is, including any comments) by the programs, this also makes any LOG file self-documenting.

Note that, unlike TABLO Input files (see section 4.1.4 of GPD-2), a comment in terminal input does not need a finishing !. Each comment finishes at the end of the line it starts on (though it can be continued by putting ! at the start of the next line).

When the program is expecting character input (such as a file name or choice of one of a set of options), it ignores any line with ! in the first column. But if you put one or more spaces at the start of the line and then '!', the program treats this as input of one or more blank characters as in

```
 ! default program options
```

If you actually want an exclamation mark (for example, in the verbal description), just put in a pair !!, which will be treated as one and the rest of the line will still be read.

## *5.3 Interactive and Batch Operation, Stored-input and Log Files*

By default all GEMPACK programs operate in an interactive mode in which the program prompts you for information and you respond by typing at the terminal. If your input is judged by the program to be invalid, the program offers an explanation and then gives you a chance to input a different response.

This is most appropriate when the program in question requires only a relatively small amount of user input. When large amounts are required, responding at the terminal can be time-consuming and error prone. Equally importantly, it can then be difficult (perhaps impossible) to reproduce your results or be sure after the event as to exactly what your results mean. For these reasons it is often important to be able to run the programs by first preparing (on a Stored-input file) all the input required and then using this file to run the program. We refer to this as running the program in **batch mode**.

GEMPACK contains a facility (independent of the operating system) whereby any of its programs can be instructed to take their input from a **Stored-input file**. This is done via the options which are presented at the start of the run of every GEMPACK program. When you start a program running you can select one of several options including the following:

```
      ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        GEMPACK OPTIONS
                  ( --> indicates those in effect )

     BAT Run in batch          STI Take inputs from a Stored-input file
     BPR Brief prompts         SIF Store inputs on a file
     LOG Output to log file    ASI Add to incomplete Stored-input file

Select an option   :  <opt>  Deselect an option      : -<opt>
Help for an option : ?<opt>  Help on all options     : ??
Redisplay options  : /       Finish option selection : Carriage return

Your selection >
```

**Options Screen for GEMPACK Programs**

In this section we discuss the options STI, LOG, SIF, ASI and BAT. We discuss option BPR and two options (**DRO** and **DRE**) which are not shown on the Options Screen (see above) in section 5.4.

To take inputs from a Stored-input file, respond (upper or lower case is fine)

```
sti
```

Then you will be asked for the name of the Stored-input file. Once you have entered this, the program takes all of its input from this file and only returns control to the terminal once it has finished running.

Another way of taking inputs from a Stored-input file is to put the name of that file on the command line as in, for example,

```
modhar  -sti  sjdat2.sti
```

This option is available on all machines (Windows PCs or Unix machines). This is equivalent to first responding "sti" and then giving sjdat2.sti as the name of the Stored-input file. See section 5.5 below for more details. [On DOS PCs or on Unix machines, you could use input redirection such as "modhar < sjdat2.sti" to run the program MODHAR taking inputs from file sjdat2.sti. However we strongly recommend that you use the GEMPACK "-sti" option in preference to this. Our reasons for giving this advice can be found in section 5.5 below.]

As you can see from the menu above, GEMPACK programs also offer you an operating-system-independent way of creating a Log file of your run. Just select option

```
log
```

after which you will be prompted for the name of the Log file. If you are running the program interactively, screen output from the program will come to the terminal screen as well as going to the Log file. Note that GEMPACK programs always echo your input so that this will also show on the Log file. If you are also taking your input from a Stored-input file, you can select option 'log' first and then option 'sti' second (or, alternatively, put 'log' followed by the Log file name at the start of your Stored-input file). In this case where both **log** and **sti** are chosen, you also must choose whether output from the program will only go to the Log file, or to both the Log file and to the screen. Full details are given in section 5.3.3.

Note that GEMPACK provides a way of specifying Stored-input file names and Log file names on the command line under DOS and Unix – see section 5.5.

When a program requires large amounts of input it can be difficult to anticipate the order of the responses required, making the preparation of a Stored-input file difficult. GEMPACK provides two options to assist you in this.

- Firstly there is the option

  ```
  sif
  ```

  which stores your inputs on a file (which can be reused as a Stored-input file) as you run the program interactively. Hands-on examples using 'sif' are given in sections 3.8.2 and 4.3 above.

- Secondly there is the option

  ```
  asi
  ```

  which lets you add to an incomplete Stored-input file. In this case the program takes its inputs from the Stored-input file you specify and then, when it comes to the end of this file, transfers control to you so you can continue running the program interactively. While this is going on, all your interactive responses are added to the Stored-input file. (Suppose, for example, you have a Stored-input file that runs MODHAR to make several changes to the data on a Header Array file and you wish to change one of these modifications. Just edit this file to remove the part you want to change and all that follows. Then run the program under option 'asi'. When it comes to the part you want to change, you will be able to add it interactively and have your responses recorded on the new Stored-input file.)

When you are running from a Stored-input file (or, more generally, in any form of batch mode), you are not in a position to correct input the program judges to be invalid. It is appropriate for the program to stop with an error message whenever invalid input is encountered (rather than running to completion and possibly producing results which may be quite different from what you wanted). All the programs stop after invalid input if any of the options 'sti', 'sif' or 'asi' have been selected. If you are running in batch mode in some other way (perhaps operating-system-dependent) you will probably want the program to stop if it encounters invalid input. To achieve this you should select the option

**`bat`**

when starting the run. (If you select option 'sti', option 'bat' is selected automatically. Option 'bat' is also selected automatically if you select option 'cmf' when running SAGEM or a TABLO-generated program. But if you are running in batch mode via an operating-system-dependent means such as DOS or Unix redirection of input, 'bat' is not selected automatically; you must select it if you want its properties.)

### 5.3.1 Invalid Input When Using Options 'sif' or 'asi'

When you are storing inputs on a file (option 'sif') or adding to an incomplete Stored-input file (option 'asi'), the file you are creating is intended for use as a Stored-input file (under option 'sti') and so must not contain invalid input (as explained above).

Accordingly, when you select either of the options 'sif' or 'asi', option 'bat' is also selected automatically. This means that, if you enter input judged by the program to be invalid, the program will stop with an error (probably "Unexpected input in batch mode"). To continue, proceed as follows.

1. Edit the Stored-input file partly created, taking out the last line (the one containing the invalid input).
2. Rerun the program this time selecting option 'asi' (even if you were previously running under 'sif'). When the program reaches the place just before it stopped, it will transfer control to you and you can give a valid response this time and continue running it.

### 5.3.2 Differences Between Batch and Interactive Program Dialogues

In this subsection, batch mode means once one of the GEMPACK options 'bat', 'sti', 'sif' or 'asi' has been selected. (It does not include cases such as DOS redirection of input when 'bat' is not included in the Stored-input file.)

In most cases the dialogue between you as user and the program is exactly the same whether you are running the program interactively or in batch mode. However the dialogue is different in a small number of clearly identified cases, all of which follow the idea that, in batch mode, the programs do not ask you to confirm something that has been set in train by your previous responses. The main examples you should be aware of are as follows.

- In MODHAR once you have given the new history information, you will not be asked in batch mode whether this is what you want.
- In the condensation phase of TABLO, if the coefficient of the substituted variable is a complicated expression, you will not be asked in batch mode whether you are sure this coefficient will never be zero.

- In the condensation phase of TABLO, if you are omitting several variables, you will not be asked in batch mode to confirm that you want to continue with this omission.[113]

### 5.3.3 Terminal Output and Log Files

When output is going to a Log file and input is being taken from a Stored-input file or a GEMPACK Command file, you may or may not want output also to go to the terminal. GEMPACK allows you to specify this as described below.

1. If, during the Options selection, you select option **log** and later select option **sti** or option **cmf** [114], before you are asked for the name of the Stored-input file or Command file to take input from, you will be asked if you want output to go to the terminal as well as to the Log file. (The default response is **B** meaning output to both the terminal and log file.)
   Hence the responses you give (for **sti**) should be:

```
log    ! Output to go to Log file
<name of Log file>  ! specify the name of the Log file
sti    ! Use Stored-input file
B      ! B (both terminal and Log file) or L (log only)
<name of STI file>
```

2. If, during the Option selection, you select option **sti** and later select (in your Stored-input file) option **log**, before you are asked for the name of the Log file you will be asked if you want output to go to the terminal as well as to the Log file. (The default is again **B** meaning output to both.) Hence in your Stored-input file you should have the following lines

```
log    ! Output to go to Log file
B      ! B (both terminal and Log file) or L (log only)
<name of Log file>  ! specify the name of the Log file
```

3. If, during the Options selection, you select option **cmf** and have previously not selected option **log**, then you can direct output to a Log file by putting the command

```
log file = <filename> ;
```

   or

```
log file = yes ;
```

   in your Command file. In this case, by default, output will also go to the terminal. However, if you put either of the commands

```
log only = <filename> ;
log only = yes ;
```

   into the Command file, this will suppress terminal output and output will go only to the Log file. See section 2.5.2 of GPD-3 for details about the statements

```
log file|only = yes ;
```

---

[113] Another case is as follows. [But since this only occurs if you run GEMSIM or a TABLO-generated program interactively (which we recommend you never do), you can probably ignore it.] In TABLO-generated programs or GEMSIM, when the closure choice is begun, if you respond 'g' to give up, you will not be asked to confirm this when in batch mode.

[114] Option **cmf** is the option which indicates that you wish to take input from a GEMPACK Command file (see sections 2.4.6, 2.4.7, 2.6.2, 2.6.3 and 2.12.2 above). It is only available when running SAGEM, GEMSIM or a TABLO-generated program.

### 5.3.4 LOG Files and Early Errors

When you select option LOG, the log file records everything echoed to the screen (even that which occurred before you selected this option[115]). In particular, if you are using a Command file, the echoing of this file is captured in the LOG file.

If you are using a Command file and a syntax error occurs while the program is reading this file, the program produces a LOG file which you can look at to see the error. The program makes up a name for this file (it will be something like "GPXX7.LOG") and tells you this name at the end of the run. This LOG file is created even if you don't have a statement "log file = ... ;" in your Command file. This LOG file helps in identifying errors in the Command file (see section 3.9.2).

## *5.4 Other Program Options*

### 5.4.1 Other Options Common to All Programs

**Option BPR**

Another standard option common to all GEMPACK programs is

`bpr`

for selecting brief prompts. This affects the prompts in situations where sets of variables are being chosen (such as in SAGEM, GEMSIM or TABLO-generated programs), and in MODHAR, as well as in a few other places. You might like to select this option once you are familiar with the various operations of the program in question.

**Options DRO and DRE**

You can ask all GEMPACK programs to delete a specified file either if the program completes successfully, or if it stops with an error. This might be useful when GEMPACK programs are being run from DOS BAT files or via other programs (such as Borland Delphi).

The two options are[116]

**dro** Delete specified file if program Runs Ok
**dre** Delete specified file if program stops with an Error

You can use these options as part of the options interaction with any GEMPACK program, though, at present these options do not appear on the Options screen (see section 5.3 above).

After you type in either of these options, you will be asked for the name for the specified file to delete. **This file must exist before you run the program**.

An example in a DOS BAT file will make this clear.

---

[115] This early output was not recorded in the LOG file in Release 5.1

[116] These options were introduced in Release 8.0.

**Example DOS BAT file**

```
REM Create error test file sjlb.err (contains just one line)
echo Error running SJLB > sjlb.err

REM Write Stored-input file SJLB.STI
echo dro > sjlb.sti
REM File SJLB.ERR will be deleted if program runs ok
echo sjlb.err >> sjlb.sti
echo cmf >> sjlb.sti
echo sjlb.cmf >> sjlb.sti

REM Run the program
sj.exe –sti sjlb.sti
if exist sjlb.err goto error
REM Has run successfully here. Continue
REM Do other things
goto done

REM Error exit
error:
echo .
echo Error running SJLB
echo .
done:
```

If the file SJLB.ERR exists after SJ has run, there must have been an error.

### 5.4.2 Options Specific to Different Programs

Some GEMPACK programs have other options (which vary from program to program). The meaning of these options should be clear from the brief description of the option (at least once you are familiar with the operation of the program in question). You can obtain Online Help screens about these options when running the program interactively from the options Menu. If you respond '**??**' you get information about all options. If you respond with a '**?**' followed by the 3-character abbreviation of an option, you receive information about that option. For example to get help on option LOG, type '**?LOG**'.

Options for GEMSIM and TABLO-generated programs are documented in chapter 14 of GEMPACK document GPD-3. Options for GEMPIE and some of the options for other GEMPACK programs are documented in GPD-4.

The method of selecting options is the same for all programs (and the same whether the option is one of those common to all programs or specific to just one program).

- You select options by typing in their 3-character abbreviation, for example type **LOG** to select the option LOG.
- You can deselect an option by typing a minus sign '-' followed by its 3-character abbreviation, for example type **-LOG** to deselect LOG.
- You can display the current options selected by typing "/".
- Once you have finished option selection you continue to run the program proper by entering a carriage-return. In fact this is usually all you will do since the default options are the most commonly used.

## *5.5 Specifying Various Files on the Command Line*

The command

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

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

You can also use **`-sti  <STI-file-name>`** to specify a Stored-input file for any program. For example, the command

modhar -sti modsj.sti

will run MODHAR taking inputs from the Stored-input file MODSJ.STI. [See the end of Example 1 in section 4.3.]

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

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

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

For example, the command

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

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

We believe that the **`-sti`** option is more robust than input redirection "<" under Windows NT, 2000 and XP in many circumstances. Indeed, we recommend that you never use input redirection for running GEMPACK programs on Windows PCs or Unix machines. [If you do, the programs do not know that you are using a Stored-input file and unexpected results may be produced.]

The command line options described above are available on Windows PCs (under DOS) and on Unix machines.[117]

If you are specifying a Command file name, STI file name or LOG file name on the Command line, enclose the name in double quotes "" **if it contains a space** as in, for example,

```
gemsim  -cmf  "c:\my  sj\my  sjlb.cmf"  -log  "c:\my  sj\my  sjlb.log"
```

### 5.5.1 Command Line Option "-los" (Alternative to "-lon")

With Release 6.0, it was possible to use "-lon" on the command line to specify that output is to go only to a log file whose name is specified after "-lon".

In Release 7.0 (or later), the command line option "-los" is an alternative to "-lon". In each case, output goes only to the log file whose name is specified on the command line just after "-los" or "-lon". The difference is that with "-lon" there is a little terminal output at the start and end of the run, whereas with

---

[117] In Release 6.0 of GEMPACK, the command line options were only available under DOS. They were made available on Unix machines in Release 7.0.

"-los" there is none. [Actually there may be terminal output if there is an error in the command line – for example if the Stored-input file specified after "-sti" does not exist. But normally there is no terminal output when "-los" is used.]

To see the difference between "-lon" and "-los" make a Stored-input file for say MKHAR and try the command

```
mkhar –sti mkhar.sti –lon mkhar.log
```

and then try it with "-los" instead.

### 5.5.2 Command Line Option "-lic" to Specify the GEMPACK Licence

The command line option "-lic <licence_file_name>" can be used to specify the name and location of the GEMPACK licence.[118] For example, the command

```
tablo  -lic  e:\test\gplic.txt
```

will run the program TABLO and treat file e:\test\gplic.txt as the GEMPACK licence.

When "-lic" is used on the command line, the usual procedure for finding the GEMPACK licence is bypassed and the file which follows "-lic" on the command line is used as the licence.

### 5.5.3 Abbreviations for LOG File Name on Command Line

In the notes below, everything said about **`-log`** applies equally to **`-lon`** and **`-los`**.[119]

**LOG File Name Based on STI File Name**

If you specify the STI file name on the Command line and you want the LOG file to have the same name except for the suffix, you can use the following abbreviations to shorten the command line.

**`-log %sti`** which means that the LOG file name is the same as that of the STI file except that the suffix of the LOG file is **'.log'**.

**`-log #sti.lox`** which means that the LOG file name is the same as that of the STI file except that the suffix of the LOG file is **'.lox'** (the part after **#sti**).

For example,

```
mkhar –sti c:\temp\mkhar2.sti –log %sti
```

means that the LOG file is c:\temp\mkhar2.log

```
mkhar –sti c:\temp\mkhar2.sti –log #sti.log3
```

means that the LOG file is c:\temp\mkhar2.log3

```
mkhar –sti c:\temp\mkhar3.sti –lon %sti
```

means that the LOG file is c:\temp\mkhar3.log

You can only use these abbreviations %sti and #sti if the "-sti" part appears in the command line before the "-log" part.

---

[118] This was introduced in Release 7.0.

[119] Abbreviating the log file name using % or # was introduced in Release 8.0 of GEMPACK.

**LOG File Name Based on Command File Name**

If you specify the Command file name on the Command line and you want the LOG file to have the same name except for the suffix, you can use the following abbreviations to shorten the command line.

**`-log %cmf`** which means that the LOG file name is the same as that of the Command file except that the suffix of the LOG file is **'.log'**.

**`-log #cmf.lox`** which means that the LOG file name is the same as that of the Command file except that the suffix of the LOG file is **''lox'**(the part after **#cmf**).

For example,

```
gemsim –cmf c:\temp\gemsim2.cmf –log %cmf
```

means that the LOG file is c:\temp\gemsim2.log

```
gemsim –cmf c:\temp\gemsim2.cmf –log #cmf.log3
```

means that the LOG file is c:\temp\gemsim2.log3

```
gemsim –cmf c:\temp\gemsim3.cmf –los %cmf
```

means that the LOG file is c:\temp\gemsim3.log

You can only use these abbreviations %cmf and #cmf if the "-cmf" part appears in the command line before the "-log" part.

In the abbreviations described above, note that **%** means that the LOG file suffix will be **.log** while **#** means that the LOG file suffix is specified after the #sti or #cmf part.

## *5.6 Fortran Compilers and Memory Management*

GEMPACK main programs are written in Fortran. Fortran is used because simulations involve extensive numerical calculations using large arrays of real numbers. The Fortran programs are compiled and linked to the GEMPACK library to make executable images when the Source-code version of GEMPACK is installed. Alternatively these executable images are supplied as part of the Executable-image versions of GEMPACK.

Fortran has two well-defined standards, the old Fortran 77 standard and the newer Fortran 90 standard, and can be used on various computers.

Compilers used with Source-code GEMPACK must be the more recent Fortran 90 compilers.[120] This applies to Windows PCs and on Unix machines.

Fortran 90 provides greatly improved memory management compared to Fortran 77. Programs are easier to use because the program parameters do not need to be adjusted to accommodate large models when using Fortran 90 GEMPACK programs.

On the PC, the Lahey Fortran compilers **LF90** and **LF95** (both Fortran 90 compilers) are the only compilers supported for use with the Source-code version of GEMPACK.[121] Executable-image versions of GEMPACK on Windows PCs are made using the Fortran 90 compiler LF90.

Chapter 13 of GPD-3 contains extra information on memory management for GEMSIM and TABLO-generated programs.

---

[120] This applies to Release 8.0 and later. In Releases 6 and 7, both Fortran 77 and Fortran 90 compilers were supported. Before Release 6, only Fortran 77 compilers were supported.

[121] Prior to Release 8, the Lahey Fortran 77 compiler F77L3 was also supported.

## 5.7 Error Messages

When a GEMPACK program encounters an error it regards as fatal, it gives some explanation and then stops with a trace-back which shows the error message and the subroutines active when the error was encountered. Usually you can concentrate on the explanation and disregard the trace-back. (You should take note of the trace-back information if you think the fatal error indicates a bug in the program.) An example is given below.

```
    (ERROR RETURN FROM ROUTINE: ANSCK)
   (E-Unexpected choice in batch)
   (ERROR RETURN FROM ROUTINE: CHSSL)
   (ERROR RETURN FROM ROUTINE: SAGEM)
```

(The traceback shows that the error "E-Unexpected choice in batch" occurred in subroutine ANSCK which had been called by subroutine CHSSL which had been called in turn by the main program SAGEM.)

GEMPACK programs contain many internal cross-checks to guard against coding errors. If one of these fails, you will see the following message.[122]

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A fatal error has occurred while running this program.
This is probably the result of an internal program error.
Please notify the suppliers of the code about this error.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

If this happens, it probably indicates a bug in one of the GEMPACK routines or programs. Please record all relevant details (such as the program you are running, the inputs you made, which files were accessed) and preserve copies of all files (including Stored-input and Command files) **exactly as they were** at the time of the error. Then report this information to us at the Impact Project. We will attempt to reproduce the error and then fix the bug (if indeed it is a bug). In order to do this, we may need to ask you to send us copies of the relevant files, so please keep the copies until we notify you that they are no longer needed.

Information about errors that may occur while running GEMSIM or TABLO-generated programs can be found in chapter 15 of GPD-3.

[122] The actual message, which is longer that the one shown here, includes email addresses to use to send the files to us.

## 5.8 GEMPACK File Types, Names and Suffixes

GEMPACK programs use a number of different file types to communicate between themselves and they produce a number of different types of files containing results for you.

### 5.8.1 Files with System-determined Suffixes

Normally you choose the names of these files. However, some types of files must be given **system-determined suffixes**, while you are able to choose the suffix for other types of files.

For example, GEMPACK requires that TABLO Input files have suffix **.TAB** (or lower case **.tab** on Unix machines). Below we list the most important type of files with system-determined suffixes.

**Table 5.8.1: File Type with System-Determined Suffixes**

| File type | Suffix[123] |
|---|---|
| TABLO Input file | .TAB |
| TABLO Information file | .INF |
| Solution file | .SL4 |
| GEMPIE Print file | .PI5 |
| Equations file | .EQ4 |
| GEMSIM Auxiliary Statement file | .GSS |
| GEMSIM Auxiliary Table file | .GST |
| TABLO-generated Auxiliary Statement file | .AXS |
| TABLO-generated Auxiliary Table file | .AXT |
| TABLO-generated program (Unix) | .f |
| TABLO-generated program (otherwise) | .FOR |
| Environment file | .EN4 |
| Model Information file | .MIN |
| Solution Coefficients file | .SLC |
| Extrapolation Accuracy file | .XAC |

Whenever a program asks you for the name of any of these files with system-determined suffixes, you should never include the suffix in your input (since the program will add the suffix automatically). For example, in a Command file put

```
Solution file = sjlb ;          ! Correct
```

rather than

```
Solution file = sjlb.sl4 ;      ! Incorrect
```

### 5.8.2 Suffixes of Other Files

Even though GEMPACK does not force you to use specific suffixes for other files, there are several different types of files where it has become customary amongst experienced GEMPACK users to use certain suffixes. Examples are in the table below.

[123] On Unix machines they are all lower case.

**Table 5.8.2: Commonly Used Suffixes for Other Files**

| File type | Usual Suffix(es)[124] |
|---|---|
| Header Array data file | .DAT, .HAR |
| Text data file | .DAT, .TXT |
| Updated Header Array file | .UPD, .HAR |
| Command file | .CMF |
| Stored-input file | .STI, .INP |
| Spreadsheet Mapping file | .MAP |

In some cases there are considerable advantages from using these "usual" suffixes. For example,

- some Command file syntax is only available if you use suffix .CMF (see section 2.5 in GPD-3).
- when you open a Header Array file with ViewHAR, by default it only shows you files with suffixes .DAT, .HAR or .UPD.

Many of the GEMPACK programs suggest suffixes for output files they create. For example, SEEHAR often suggests suffix .SEE for its output files. We suggest that you go along with these suggestions unless you have good reasons to do otherwise.

### 5.8.3 Files - Binary or Text?

There are two basic file types on all computers - text files (which are sometimes called ASCII files) and binary files. The former can be printed or edited while the latter cannot be. GEMPACK uses several files of each type, as discussed below.

Below is a list of common GEMPACK files (some of which you may not meet for some time) with information as to whether they are binary or text files.

Several GEMPACK programs output a text file which we often call a **Print file** since it can be sent to a printer.

Base data for models can be held on text or binary files. When binary files are used, they must be GEMPACK Header Array files (see chapter 4), where each array of data is identified via its 4-character header. Text data files used in GEMPACK must follow a standard format which was introduced in chapter 4 and is documented fully in chapter 6 of GPD-4.

| File Type | Binary or Text |
|---|---|
| TABLO Input file | text |
| TABLO Information file | text |
| Solution file | binary |
| GEMPIE Print file | text |
| Equations file | binary |
| Extrapolation Accuracy file | text |
| TABLO-generated program | text |
| GEMSIM Auxiliary Statement,Table files | binary |
| TABLO-generated Auxiliary files | binary |
| Environment file | binary |
| LU file | binary |
| Base data files | can be binary or text |
| Updated data files | can be binary or text |
| GEMPACK Command files | text |
| Stored-input files | text |
| Solution Coefficients file | binary |

[124] On Unix machines they are usually all lower case.

### 5.8.4 Why So Many Files?

We have listed many of the important files, and discussed their roles, in section 2.15 above.

Some files are created to facilitate communication between different GEMPACK programs. [For example, GEMSIM Auxiliary files allow communication between TABLO and GEMSIM.]

Other files are created for users to look at or use. Some contain information which is important in reporting simulation results while others may allow experienced modellers to carry out tasks more efficiently. For example, when GEMSIM or a TABLO-generated program runs, it may produce various files listed below.

- The **updated data files** provide information about the simulation and can be used as the starting point for other simulations (see section 2.9).
- The **Equations file** can be used as the starting point for Johansen simulations (see section 2.12.1 above).
- The **Extrapolation Accuracy file** can be used to tell if the solution produced is sufficiently accurate for your purpose (see section 2.13.3). (If not, you may need to re-run it with more steps.)
- An **Environment file**. Saving an Environment file makes it easy to run a different simulation with the same closure: you simply give the name of the Environment file instead of having to re-specify which of the variables are exogenous or endogenous. Indeed, if you just want to make a small change in the closure, it is still easiest to start from an existing Environment file. More information about this is given in section 5.2.2 in GPD-3.

## *5.9 Allowed File and Directory Names*

What file and directory names are allowed in conjunction with GEMPACK programs differs widely between different machines and operating systems. It also depends on the Fortran compiler used.

In this section we give detailed information about this. The information in this section is rather technical. Beginners should skip it. Cross reference to this section will be made from several different parts of the GEMPACK user documentation.

We begin with some general points which apply to all operating systems and compilers.

- All letters A-Z and/or a-z and all digits 0-9 are allowed in file names on all machines. However, the length of such file names may be limited (see later in this section). If you stick to such simple and practical names, you will not need to know many of the details which appear below.
- GEMPACK programs never access or produce files whose names end with one of more spaces. For example, if you try to create a file called "xx.out " the trailing space will be omitted and the file "xx.out" will be created.
- Directory names are usually restricted in the same way that file names are. Even if you are trying to create a file with a file name which is allowed with the particular version of GEMPACK you are using, it may not be possible to create the file if you are trying to do so within a directory whose name is not an allowed one. For example, if you are working with LF90 MODHAR, you may ask it to create a file called SJ2.DAT. However if you are running inside the directory containing a character that the Fortran compilers cannot handle [for example, "C:\temp=files\sj"] you would not be able to create the file since the "temp=files" part of the directory name contains the character "=" which is not allowed with LF90 GEMPACK programs (see section 5.9.3 below). Similarly Chinese or Scandinavian characters in a directory name may cause problems.

On Windows PCs, what file names are allowed depends on the compiler used to make the program which is running. [See the relevant sections below.]

Most Unix compilers are able to handle long filenames or filenames with embedded spaces. Consult your computer manager to find out.

### 5.9.1 Are File Names Case Sensitive ?

On Unix machines, file names are case sensitive. So, for example, the file sj.dat is different from the file sj.DAT.

On other machines (including Windows PCs), file names are not case sensitive. So, for example, there is no distinction between the file names sj.dat and sj.DAT. Thus, if you create a new file called "sj.dat" you will, in the process, delete any existing file called SJ.DAT or sj.DAT.

### 5.9.2 File Names on Windows PCs – Programs Made with LF90 or LF95

Programs distributed with the Executable-image and Demonstration Versions of GEMPACK are made using LF90 so that this section applies to these versions of GEMPACK.

Before Release 7.0 of GEMPACK,

> file names were restricted to the original DOS 8.3 format (8 characters plus a full stop and a 3-character suffix).[125] The allowed characters were limited also to those currently allowed under F77L3 – see section 5.9.3 below. In particular, spaces were not allowed in file or directory names.

With Release 7.0 (and later),

> it is possible to use **long file names and directory names**. Also file and directory names can include spaces and most other characters allowed in file names with the Windows operating systems Windows 95, 98, ME, NT, 2000 and XP.[126]
>
> A further restriction is that, with LF90 programs, characters ";" and "=" are not allowed in file or directory names. [These characters are allowed with LF95 programs.]
>
> In addition the GEMPACK Windows programs (WinGEM, ViewHAR etc) do not reliably support file names containing "%" or "#" so we advise you to not to use these characters in any file names.
>
> GEMPACK programs do not reliably support file and path names containing Chinese or other non-English characters. Even if your file names are in English characters, a directory name containing non-English characters can cause problems. We apologise for this inconvenience.
>
> We expect many users will want to use long file names to make these names more self-documenting. But even then, we advise you to stick to letters, digits, spaces, "-" and "_" and to avoid other characters; this is a simple and practical way of avoiding possible complications.

### 5.9.3 File Names on Windows PCs – Old Programs Made with F77L3

The Lahey Fortran 77 compiler F77L3 is no longer supported. However you may have old TABLO-generated programs made with F77L3 so this section is included as a reminder of the conditions under which these old programs will run.

The Lahey Fortran compiler F77L3 can not handle more than the original DOS 8.3 format (8 characters plus a full stop and a 3 character suffix) for file names. Directory names are limited to at most 8 characters.

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

The only characters apart from letters and digits allowed in file or directory names are "_" (underscore), "-" (hyphen) and "~" (twiddle). The characters ":", "\" and "." have their usual DOS meanings.

---

[125] In fact if you use version 4.05i or later of LF90 in conjunction with Release 6.0 of GEMPACK, long file names (as described with Release 7.0) are available. [This is because the default compiler option changed with version 4.05i of LF90 so that winconsole images are created by default.] However spaces are not allowed in file or directory names and the allowed characters are restricted to those allowed currently with F77L3 programs.

[126] File and directory names on these systems cannot include any of the following characters:
**\ / : * ? " < > |**

Note that GEMPACK programs from Release 6.0 or earlier made with F77L3 are prone to giving misleading error reports if you specify a file name with more than 8 characters before the suffix. For example, if you ask such a program to open the file **verylongname.dat**, the program truncates the "verylongname" part to 8 characters and so actually tries to open the file "verylong.dat". If it does not find such a file "verylong.dat" it mystifyingly reports that the file "verylongname.dat" (with all 12 letters) does not exist, even if it does exist. On the other hand, if a file "verylong.dat" also exists, the program will open that file (instead of "verylongname.dat"). Release 7.0 (or later) GEMPACK programs compiled with F77L3 attempt to provide their own error trapping to give reliable error messages in cases like those described above. In particular, they should tell you that you have more than 8 characters before the suffix.

### 5.9.4 File Names Containing Spaces

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

  **solution file = "my sj" ;**

  See section 2.4.2 of GPD-3 for details.

- If you are running a program interactively, you must not add quotes "" when specifying file names containing spaces. Similarly you should not add quotes when specifying such file names in Stored-input files. For example, if you are running SEEHAR interactively to look at the data in file

  **c:\my  sj\sj.dat**

  you respond exactly like that (rather than "c:\my  sj\sj.dat") when you are asked for the file name.

- If you are specifying a Command file name, STI file name or LOG file name on the Command line (see section 5.5), enclose the name in double quotes "" if it contains a space as in, for example,

  **gemsim -cmf "c:\my sj\my sjlb.cmf" -log "c:\my sj\my sjlb.log"**

### 5.9.5 Characters in Stored-input Files and Command Files

TAB characters and other control characters (ASCII characters 0-31 and 127) can cause problems in Stored-input files and Command files. In Release 8.0, TAB characters are replaced by a single space. Most control characters are replaced by a single space but will cause a warning message. The program will stop with an error message if it finds a Control-Z character before the end of the file if there is text after it, or if there are two or more in the file.

There is no testing for other characters (ASCII characters 128-255) but these would cause problems if used in file names – see section 5.9.2.

# CHAPTER 6

# 6. A Guide to the Rest of the Documentation

There are currently seven GEMPACK documents which are referred to by their GPD numbers:[127]

GPD-1, GPD-2, GPD-3, GPD-4, (no GPD-5), GPD-6, GPD-7, GPD-8.

Their full titles are given in chapter 9.

This document GPD-1 *An Introduction to GEMPACK* is the first of the GEMPACK documents. It is an introduction and broad overview of the GEMPACK framework as a whole. We assume in the other GEMPACK documents that you have read or are familiar with the material in chapters 2, 3 and 4.

GEMPACK documents GPD-2, GPD-3 and GPD-4 contain the full details relating to using particular programs and fine points of syntax. The material in these documents was rewritten and reordered for Release 7.0 as follows:

- Material related to the program TABLO and to TABLO Input files is in GPD-2: *TABLO Reference.*
- GPD-3: *Simulation Reference: GEMSIM, TABLO-generated Programs and SAGEM* contains information about writing Command files and how to run simulations. In particular chapter 18 of GPD-3 summarises Command file syntax. GPD-3 also contains material on run-time errors and memory management.
- GPD-4 *Useful GEMPACK Programs* contains details about all the other GEMPACK utility programs such as MODHAR, SLTOHT and GEMPIE. Chapter 6 contains details on the syntax of GEMPACK Text files. There is some material on Windows programs WinGEM, ViewHAR, ViewSOL, TABmate, RunGEM, RunDynam and AnalyseGE but these program also have their own on-line help files.

Installation instructions and machine-specific information for PC computers are given in GPD-6 and GPD-7.

- Source-code GEMPACK installation is in GPD-6 *Installing and Using the Source-Code Version of GEMPACK on Windows PCs with Lahey Fortran,* and
- GPD-7 *Installing and Using the Executable-Image Version of GEMPACK on Windows PCs* is for Executable-image versions of GEMPACK.
- Users of GEMPACK on Unix and other machines should also be able to obtain machine-specific information from their GEMPACK Managers. (These are brief documents not included in the GPD series.)

GEMPACK document GPD-8 *Getting Started with GEMPACK: Hands-on Examples* suggests hands-on computing (on Windows machines or Unix and other machines working from a command prompt) based on the example files which are supplied with GEMPACK. These examples files which are usually in the Examples subdirectory of your GEMPACK directory are a reference in themselves and provide examples of many points of syntax used in TABLO Input files and Command files.

---

[127] 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".

GPD-8 should probably be your next port of call after you have read GPD-1. Documents GPD-2 to GPD-4 are essentially reference documents, not designed to be read sequentially, but rather used as needed. In particular, they contain detailed Index entries (at the back) and contain many cross references designed to assist you find the information you need.

## *6.1 Other Relevant Publications*

Background about the rationale for the development of GEMPACK can be found in Pearson (1988) and Codsi and Pearson (1988); these both describe GEMPACK before multi-step solutions were incorporated. An earlier version of just the Johansen simulation parts (SAGEM and GEMPIE) of GEMPACK published for use with DPPW is Pearson (1992).

An overview of GEMPACK (specifically Release 5.1) can be found in Harrison and Pearson (1996). When you report model results obtained using GEMPACK, this is probably the most appropriate general GEMPACK reference to cite.

The procedure for implementing and solving intertemporal models via GEMPACK is given in Codsi *et al* (1992).

A discussion of the pros and cons of implementing models via levels or linearized representations can be found in Hertel *et al* (1992). Suggestions about the desirability of working with mixed representations of models (that is, some levels and some linearized equations) can be found in Harrison *et al* (1993a) and (1993).

Features of GEMPACK which are useful in dealing with large models, especially multi-regional or intertemporal ones, are discussed in Harrison *et al* (1996).

Kohlhaas and Pearson (2002) provides an introduction to GEMPACK for GAMS users.

# CHAPTER 7

# 7. Recent GEMPACK Releases

In this chapter we record the new features of the most recent two releases of GEMPACK before the current release, namely Release 6.0 (October 1998) and Release 5.2 (September 1996). We also list the GEMPACK documentation which was current for those releases.

The information in this chapter is included mainly for reference purposes. It may also be useful for readers who have jumped to Release 7.0 from Release 5.2 of earlier.

## *7.1 Release 7.0 (October 2000) - Changes From Earlier Versions of GEMPACK*

The most important new features in Release 7.0 were:

- the ability to calculate subtotals. In particular, this enables you to decompose the results of a non-linear simulation into the contributions made by different groups of shocks (see point 1 below).
- the Windows program AnalyseGE which assists modellers to analyse their simulation results (see point 2 below).
- the complete reorganisation and consolidation of the documentation. We hope and expect that this will assist both new and experienced users to make more efficient use of the software.

Below is a brief description of the new features in Release 7.0.[128]

1. **Subtotals** can be produced by GEMSIM and TABLO-generated programs.

   Subtotal statements in Command files produce subtotal solutions on the Solution file. The Command file syntax is the same as for subtotals produced by SAGEM.

   This allows you to decompose the results of a simulation into the effects of different groups of shocks.

   See chapter 11 of GPD-3 for more details.

2. **AnalyseGE** is a new Windows program for analysing simulation results.

   AnalyseGE is aimed at making it easy for users to calculate expressions involving Coefficients and Linear variables such as those occurring in linearized equations. AnalyseGE obtains the values of variables from a Solution file and the values of Coefficients from the associated SLC file (see point 3 below). Once the expression is calculated it is immediately available in a ViewHAR-style window. See section 2.6 of GPD-4 for more details.

[128] The text here is taken from the Release 7.0 documentation. In particular, "now" in this section means October 2000, when Release 7.0 was made available. However, cross references relate to the current (Release 8.0) versions of the GEMPACK documents.

3. **Solution Coefficients File** contains values of Coefficients.

   When TABLO-generated programs or GEMSIM carry out a simulation, by default they produce a new output file called a Solutions Coefficients file, with suffix **.SLC**, which contains the pre-simulation value of all Coefficients defined in the TABLO Input file. This .SLC file is a Header Array file. The long names on each array indicate the name of the Coefficient whose values are held at each header. See section 8.4 of GPD-3 for more details.

4. The **RunDynam** Windows program is available for **recursive dynamic models** such as MONASH and the dynamic version of GTAP.

   RunDynam (and various model-specific versions such as **RunMONASH** and **RunGDYN**) are available for recursive dynamic models. These interfaces are designed to remove much of the computing complexity in working with forecasts and policy simulations which run over several consecutive years. RunDynam is not supplied with GEMPACK but can be purchased separately. See section 2.7 of GPD-4 for more details.

5. **Systematic Sensitivity Analysis** can be carried out using RunGEM.

   The version of RunGEM first distributed with Release 6.0-001 of GEMPACK makes it easy for users to carry out systematic sensitivity analysis with any models implemented and solved using GEMPACK. You can compute how sensitive your simulation results are to changes in the parameters or to changes in the shocks. See section 2.5.3 of GPD-4 for more details.

6. **Set Mappings** have been extended in various ways.

   Various improvements make setting up mappings easier. You can now read part of a set mapping BY_ELEMENTS. Mapping values can be given using the values in other mappings using a FORMULA(BY_ELEMENTS) and an index-expression.

   Mappings can be written as character data using a WRITE(BY_ELEMENTS) statement.

   It is now possible to insist that a set mapping is **onto**, which means that every element in the codomain is mapped to by at least one element of the domain set.

   ViewHAR allows you to save set mappings as Header Array files or text files or to paste them into spreadsheets or TABLO Input files.

   See section 4.8 of GPD-2 for more details.

7. **Shocks** can be read **from Header Array files.**

   The values of shocks can now be read from a header on a Header Array file. [Previously, they could only be read from a text file.] The Command file syntax is as for text files except that you must add "header <header>" at the end. An example is:

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

   This can sometimes be faster than reading shocks from a text file and you can have more than one shock on the same file, held at different headers. See section 5.5.2 of GPD-3 for more details.

Program SEENV can be used to assist you to write these shock statements. See section 12.3 of GPD-4 for more details.

8. **Header Array files can be converted to Data Bases** using SEEHAR and option SQL.

   A new option **SQL** in SEEHAR enables you to produce SQL output which makes it relatively easy to add the data in Header Array files file to most databases. See section 4.1.4 of GPD-4 for more details.

9. **Long file names and spaces in file names** are allowed on Windows PCs using the Lahey LF90 or LF95 compilers.

   This means that you are no longer restricted to names of length 8 or suffixes of length 3, and you can include spaces in file and directory names, if you are working with LF90 or LF95 on a Windows PC. [This also applies to the Executable-image and Demonstration versions of GEMPACK.] See section 5.9 for more details.

10. The **LF95 Fortran compiler** is now supported on Windows PCs.

    LF95 (Lahey Fortran 95) is an alternative to LF90. See GPD-6 and chapter 13 of GPD-3 for more details.

    This has consequences for sharing Header Array files between different users since the binary files (including Header Array files) produced by LF95 are not the same as those produced by LF90 or F77L3. However GEMPACK programs have been written so that you will usually not need to be concerned about the differences, and we supply programs for converting between the two types of files. See chapter 15 of GPD-4 for details.

11. We expect to **phase out support for Fortran 77** compilers in the next release.

    Some of the new features in GEMPACK are only available with a Fortran 90 compiler. Also Fortran 90 versions of GEMPACK are much easier to use since they do their own memory management, which means that users to not have to increase program parameters. [See section 5.6 below and chapter 12 of GPD.3 for more details.]

    We strongly recommend that you upgrade to Fortran 90 if you are still using a Fortran 77 compiler. This applies to all machines including Windows PCs and Unix machines. In particular, on Windows PCs, we strongly recommend that you upgrade from F77L3 to either LF90 or LF95.[129]

12. We are no longer supporting **Windows 3.1.**

    The Windows programs (including WinGEM, GemEdit, RunGEM, ViewHAR, ViewSOL, TABmate and AnalyseGE) provided with Release 7.0 will not run under Windows 3.1. They are designed for the later Windows operating systems including Windows 95, 98, NT and 2000. See

[129] The basic version of LF95, which is called **LF95 Express**, is suitable for use with Source-code GEMPACK. At the time of writing, this costs only approximately $US200, which provides a relatively inexpensive way of upgrading for F77L3 users.

chapter 2 of GPD-4 for more details.

13. There is improved **error trapping** for invalid powers and log functions, and also for other arithmetic problems including overflow.

    For invalid powers or log functions, error messages supply details of the values and of the operation. For example, the error message may say that you are trying to raise a negative integer to a negative power, or that you are trying to take the logarithm of a negative number.

    If arithmetic overflow occurs, the software can usually tell you where it occurred (for example, in a certain formula or while extrapolating the results for a certain component of a named variable).

    Previously these errors were trapped by the operating system rather than by GEMPACK. See sections 15.5 and 15.6 of GPD-3 for more details.

14. **Range checking** can now be done in some circumstances even if not you are not using automatic accuracy.

    New Command file statements are:

    range test updated values = updated|extraploated|both|no|warn :
    range test initial values = yes|no|warn ;

    See section 6.4 of GPD-3 for more details.

15. Improvements have been made to **ViewHAR, ViewSOL and TABmate**.

    These 3 programs have been made more user-friendly in a number of small ways; for example, printing is more convenient. Other improvements include:

    **ViewHAR**: more support for working with sets and mappings, including a built-in **data aggregation facility**. You can sort across the rows of an array.
    **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. You can make charts of results and open several Solution files side by side.
    **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.

    See chapter 2 of GPD-4 for more details.

16. New TABLO statement **TRANSFER** and new Command statement **XTRANSFER**.

    When a TABLO Input file has instructions to read data from a Header Array file, there may be extra data on the file which is not read when the TABLO-generated program or GEMSIM runs. Sometimes you may want some or all of this extra data to be transferred to the new or updated Header Array files written. A new TABLO statement TRANSFER makes this easy to carry out. See

sections 3.15 and 4.12 of GPD-2 for more details.

17. New command line options **–los** and **-lic** are available.

The command line option **`-los <log_filename>`**
is an alternative to **-lon**. In each case, output goes only to this log file. The difference is that with **-lon** there is a little terminal output at the start and end of the run, whereas with **-los** there is none. See section 5.5.1 for more details.

The command line option **`-lic <licence_filename>`**
can be used to specify the name of the GEMPACK licence file. See section 5.5.2 for more details.

18. **Command line options** are now **available on most Unix machines**.

The command line options including -cmf, -log, -sti which previously were only available on PCs are now available on most Unix machines. See section 5.5 for details of these command line options.

19. **TABLO Input file and TABLO STI file are stored on Solution file**.

In Release 6.0, the TABLO Input file was stored on the Auxiliary Table (.AXT or .GST) file. The TABLO Input file is now transferred from the Auxiliary Table file (providing it is there) to the Solution file when you carry out a simulation using GEMSIM or a TABLO-generated program. Similarly, if you use a Stored-input file to run TABLO, this Stored-input file is now transferred to the Auxiliary Table file and is also transferred to the Solution file when you run a simulation.

20. **Compatibility of various pairs of files is now checked**.

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

The same time and date stamp is put on the Solution and .SLC files produced in a single run of a TABLO-generated program or GEMSIM. The program AnalyseGE uses these to alert the user if the files were not produced at the same time.

SLTOHT puts the time and date stamp from the Solution file onto the Header Array file it creates. Programs such as ViewSOL use this to decide if an existing .SOL file comes from the specified Solution file.

21. **List of exogenous/endogenous variables is shown when closure is invalid**.

If the closure specified on a Command file is not valid because of an incorrect number of exogenous variables, the program gives a list of the exogenous variables and a list of the endogenous ones (each in Command file syntax) before exiting with an error. Users may find these lists helpful in identifying variables incorrectly specified as exogenous or endogenous. We are grateful to Mark Horridge for suggesting this.

22. The **EQUATION(NONE)** statement is no longer required with Fortran 90 TABLO.

    If you are using a version of TABLO which has been compiled with a Fortran 90 compiler (this includes the Executable-Image version of GEMPACK), you do not need to include the statement

    EQUATION (NONE) ;

    (see the footnote in section 4.16 in GPD-2) in your TABLO Input file. TABLO infers this after it has done the preliminary count of the numbers of various statements (including the number of equations). [However, this statement is still accepted by Fortran 90 TABLOs.]

23. **Target shocks** are allowed for recursive dynamic models.

    This applies to recursive dynamic models such as MONASH and, in particular, to the RunDynam windows software (see section 2.7 of GPD-4). In some policy simulations, you may have a target change in a variable which was endogenous in the base case but exogenous in the policy simulation. Previously you could not set the shock for the policy run until you had seen the endogenous results from the base case (since the target could only be met by giving just the right ashock value so that the combined result of the inferred shock from the base case and the policy ashock value is the desired target) . Now you can use a "tshock" statement to hit the target. See section 5.5.4 of GPD-3 for more details.

24. **Three new functions RANDOM** (for generating random numbers), **NORMAL** and **CUMNORMAL** (relating to the normal probability distribution) are now available. See section 4.4.4 of GPD-2 for details.

25. If you use Newton's method, information about the size of the **Newton correction terms** is available even if you are not carrying out a simulation. This enables you to easily check how well the equations of your model are satisfied in the initial data base. We are grateful to Ronald Wendner for suggesting this. Details can be found in section 7.5.3 of GPD-3.

### 7.1.1 GEMPACK Documentation for Release 7.0


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, First edition, October 2000.

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

## *7.2 Release 6.0 (October 1998) and Release 6.0-001 (March 1999)*

### 7.2.1 New Features in Release 6.0 (October 1998)

The new features and enhancements of Release 6.0 (October 1998) over and above Release 5.2 (September 1996) related mainly to the following areas.[130]

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

It is now possible to ask GEMSIM or a TABLO-generated program to calculate and report pre-simulation and post-simulation levels values as well as the usual percentage change results. To do this, you may need to add appropriate information in your TABLO Input file to tell TABLO what are the pre-simulation levels values of selected linear variables. For example, in the linearized TABLO Input file SJLN.TAB for the Stylized Johansen model (see section 3.4.1), add the qualifier "(ORIG_LEVEL=DVCOM)" when declaring the variable p_XCOM, as in

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

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

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

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

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

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

[130] The text here is taken from the Release 6.0 documentation. In particular, "now" in this section means October 1998, when Release 6.0 was made available. However, cross references relate to the current (Release 8.0) versions of the GEMPACK documents.

the percentage change p_XCOM in XCOM and p_PCOM in PCOM, for each commodity. See section 4.5.5 of GPD-2 for details.

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

It is possible to declare a **specified range** within which each coefficient must stay (for example if X(i) > 0 this is written as a qualifier "GT 0" for a coefficient that should always be positive). Checks that the coefficient stays within this range are carried out at run time when you are using automatic accuracy. [If it goes outside the range during a given subinterval, the subinterval is repeated with a shorter length.] See section 6.4 of GPD-3 for details.

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

When GEMPACK programs are used in conjunction with a suitable Fortran 90 (as distinct from Fortran 77) compiler, you no longer need to increase program parameters which are too small for your model. The GEMPACK programs now use the memory management features available in Fortran 90 to allocate just the amount of memory needed to complete each task. See chapter 13 of GPD-3 for details.

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

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

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

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

Details are in section 4.6.2 of GPD-2 for details.

(5) **Extended syntax with intertemporal models** can be used which allows indices such as "t+1" to be used in sensible circumstances in which they were not allowed in Release 5.2. See section 7.4 of GPD-2 for details.

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

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

```
SET C = A union B
SET D = A intersect B
```

See section 4.6.3 of GPD-2 for details.

(7) Various **speed-ups** have been made in the way simulations using automatic accuracy or using the MA28 routines are carried. See section 8.1 of the second edition of GPD-4 (for Release 6.0) for more details.

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

When writing an array to a Header Array file, if the long name to use is not specified in the TABLO Input file, TABLO-generated programs and GEMSIM transfer the long name (if not all blanks)

from when this same data was read. The intention here is to make it easier for modellers to **preserve long names** on files when doing data organisation tasks. See section 4.10.6 of GPD-2.

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

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

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

would write the elements of the set COM to a Header Array file (the one with logical name "hafile") and to a text file (the one with logical name "textfile") respectively (assuming the relevant logical files had previously been declared to be new files of the appropriate type).
See section 4.6.6 of GPD-2.

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

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

[See section 4.6.7 of GPD-2.] This is especially useful if you want to add set and element information to a Header Array file via MODHAR (see section 5.3.4 of GPD-4).

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

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

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

See section 5.3.4 of GPD-4.

(11) **Checking set, element and coefficient information when reading data**

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

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

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

See section 5.2 of GPD-4 and section 4.4 of GPD-3 for details.

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

(a) The new option SSS allow you to make a **"short" spreadsheet** or CSV file containing no comments, blank lines or "how much data" information. See section 9.2 of GPD-4. The other new option SHK converts values read from a Solution file into "shock" statements for use in CMF files (see section 8.8 of GPD-4),

(b) The default is now for SLTOHT to use **element names** rather than component numbers in output spreadsheet files. Component numbers (the only possibility in earlier releases) can be obtained via option NEL. See sections 8.9 and 9.2.1 of GPD-4.

(c) Option SSE can now produce spreadsheet tables with row and column labels when only part of a variable is specified on the Spreadsheet Mapping file. See section 9.3 of GPD-4.

(d) Whenever option SS or SSS is used to produce a spreadsheet style output file, it is now easy to produce **tables of results** in which results for different variables are in the columns. See section 9.4 of GPD-4.

(e) The tables produced with SSE or SS or SSS options should be ideal for making graphs of results (including with intertemporal models). See section 9.5 of GPD-4.

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

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

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

(14) **Command file streamlining**

Command files list various file names such as Solution or Log file names which by convention are often the same as or similar to the name of the Command file. To facilitate this convention, and avoid unnecessary editing of files, various default file names have been introduced.
For example, if you omit the **Solution file = ...;** statement from the Command file say **sjlb.cmf**, the Solution file will be called the same name as the Command file **sjlb.sl4** by default. See section 2.5 of GPD-3 for details.

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

The DOS command

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

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

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

```
gempie   -sti  gempie.sti    -lon  gempie.log
```

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

See section 5.5 for details about these command line options **-cmf**, **-sti**, **-log** and **-lon**.

(16) **Programs RWSOL and MKSOL**

A pair of utility programs call RWSOL and MKSOL have been written to transfer Solution files from one machine to another, similar to the pair of programs RWHAR and MKHAR used to transfer Header array files. The idea was that Solution files on Unix machines could be transferred to the PC to be viewed with the Windows program ViewSOL. See section 11.2 of GPD-4 for details.

(17) **GEMPACK licence files**

SAGEM now requires a GEMPACK licence. When running with a large model, TABLO-generated programs require a GEMPACK licence; see section 1.9.6. See also section 1.9.

(18) **ViewSOL**

ViewSOL, a Windows program for viewing Solutions has been developed by Mark Horridge. See section 2.3 of GPD-4 for details. ViewSOL can also be used to view Levels results if they are present on the Solution file.

(19) **Editing mode in ViewHAR**

ViewHAR, the Windows viewer for Header Array files, has been extended to include an editing mode. Details are given in the Online help for ViewHAR. See also section 2.2 of GPD-4.

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

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

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

Details about RunGEM can be found in chapter 2.5 of GPD-4 and chapter 5 of GPD-8.

(21) **TABmate**

TABmate written by Mark Horridge is designed to be used for modifying and debugging TABLO Input files. It is aimed at the intermediate or advanced GEMPACK user and is particularly helpful for large or complex TAB files. See section 2.4 of GPD-4.

(22) **Programs ACCUM and DEVIA**

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

years of these runs into a spreadsheet file while DEVIA produces a spreadsheet file which shows the differences between the policy (deviation) run and the base case. These programs are described in more detail in chapter 10 of GPD-4.

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

Statements of the form

ashock … ;

(for example, "ashock xtax = uniform 5 ;") can be used to specify additional policy shocks with dynamic models (such as MONASH). See section 5.5.4 of GPD-3 for details.

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

It is now possible to ask GEMSIM or a TABLO-generated program to check the closure and shocks statements in any Command file. This will happen quickly, even with a large model. See section 5.9 of GPD-3 for details.

(25) **Absorption during condensation**

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

### 7.2.2 New Features in Release 6.0-001 (March 1999)

Release 6.0-001 contained several enhancements to the features in Release 6.0. These enhancements are summarised in this section.[131]

(1) **Range checking can apply when not doing automatic accuracy**

If are carrying out a simulation, range checking (see section 6.4 of GPD-3) can now apply when automatic accuracy is not being done.

The statements "range test updated values = ... ;" and "range test initial values = ... ;" are now extended to add an additional alternative value of "warn". That is, they have the form

**range test updated values = updated|extrapolated|both|no|warn ;**
**range test initial values = yes|no|warn ;**

When not doing automatic accuracy, the default value is "WARN" in each case. [When doing automatic accuracy, the default values are still BOTH and YES respectively, as set out in section 7.4.6 of GPD-3.]

When not doing automatic accuracy, if one of the values is out of range,

(i) if the relevant "range test ..." is set at "WARN", then a warning is shown for the first few instances but the simulation carries on as if these warnings were not given.

---

[131] The text here is taken from the Release 6.0-001 documentation. In particular, "now" in this section means March 1999, when Release 6.0 was made available. However, cross references relate to the current (Release 8.0) versions of the GEMPACK documents.

(ii) if the relevant "range test ..." is set at "YES", then the simulation ends with a fatal error after the value out of range is shown (in the log file). Possibly several out-of-range values of the same type (initial, updated or extrapolated) may be shown before the program stops.
(iii) if the relevant "range test ..." is set at "NO", the relevant testing is not done.

We recommend setting these to "YES" so that simulation results in which a value goes out of range are not reported. [If a value goes out of range, you can increase the number of steps or use automatic accuracy to keep the values in range.]

(a) If "range test updated values = both ;", then both updated and extrapolated values out of range are fatal errors.
(b) If "range test updated values = updated ;", then updated values out of range are fatal errors, and the software merely warns about extrapolated values out of range.
(c) If "range test updated values = extrapolated ;", then extrapolated values out of range are fatal errors, and the software merely warns about updated values out of range.

If automatic accuracy is being done, then "WARN" is a new possible value for these range test ... ;" settings. If the relevant "range test ..." is set at "WARN", a warning is shown for the first few instances out of range but the simulation carries on as if these warnings were not given. [In particular, the subinterval is not redone.]

The only difference between setting one of these to "NO" or to "WARN" is that in the latter case you see warnings in the log file.

If any range checks are set to be warnings only, a summary at the end tells how range check failures there have been (if there are any). You can check if there have been any by searching for "not in the required range" in your log file.

(2) **Assertion failures can now be warnings**

There is a new statement

**assertions = yes|no|warn ;**

available in Command files for TABLO-generated programs and GEMSIM (see section 6.3 of GPD-3).

(i) The statement "assertions = no ;" is equivalent to the statement "nas = yes ;". When this is in operation, Assertions are not checked.
(ii) The statement "assertions = yes ;" is equivalent to the statement "nas = no ;". This is the default. When this is in operation, assertion failures are treated as fatal errors.
(iii) When the statement "assertions = warn ;" is in operation, assertions are checked, but a failure results in a warning only. The first few warnings are shown in detail on the log file.

A summary at the end tells how many assertion failures there have been (if there are any). You can check if there have been any by searching for "assertion failure" in your log file.

(3) **Reporting memory used by TABLO-generated programs and GEMSIM**

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

The memory used is reported in two parts (referred to as the TGMEM1 and TGMEM2 parts). For large models, the TGMEM1 part is usually the larger of these.

Note that these reports do not include memory required for the nonzeros generated doing the LU decomposition (where memory is allocated according to the size of the parameters MMNZ and MMNZ1). The amount of memory used for these is

12*MMNZ bytes if MA48 is used,
(4*MMNZ + 8*MMNZ1) bytes if MA28 is used.

The minimum MMNZ and MMNZ1 values required are echoed at the end of the run of these programs.

With these reports you can get a rough estimate of the total memory required to run the program. To do this, add the TGMEM1 and TGMEM2 parts, add the above values for MMNZ,MMNZ1 and then add about 4-5 megabytes for other memory and for memory required for the code itself.

(4) **Programs ACCUM and DEVIA** (see chapter 10 of GPD-4)

(i) ACCUM can now produce accumulated results as well as side-by-side results. See the new options ACC,ACI.
(ii) DEVIA can now produce year-on-year differences. See the new option NAC.
(iii) DEVIA can now be told the name of a Solution file associated with the results being processed - see new option SOL. If you use this option, DEVIA will correctly process change and percentage-change variables.
(iv) The Release 6.0-001 versions of these programs run considerably more quickly than the Release 6.0 versions. Some bugs have been fixed.

(5) **New option ADD when modifying data in MODHAR**

When you modify the data on an array using MODHAR, the Release 6.0 version of MODHAR offered the options "r" (replace) or "s" (scale).
The Release 6.0-001 version of MODHAR also offers the new alternative "a" (add). With this option, the numbers entered are added to the values at the original header. [See section 3.8.4 of GPD-4.]

(6) **GAMS output from SEEHAR, TABLO-generated programs and GEMSIM**

These now add GAMS-type lines at the start and end of the files (see section 4.1.3 of GPD-4).

(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.

Enhancements (1) and (ii) should make it easier for these files to be joined together with other GAMS files.

(7) **Default for levels results and SLTOHT**

The Release 6.0 documentation omitted the fact that "NLV" is the default for SLTOHT.

(8) **"Check-on-read coefficients" statements**

The default value was not specified unambiguously in the Release 6.0 documentation. [On page 5-33 (section 5.2) of GPD-4 (second edition, October 1998), the default was stated to be "warn". In chapter 19, page 19-114, of the same edition of GPD-4, it was stated to be "no".] In Release 6.0-001, the default has been set to "no".

(9) **Slight incompatibility between Release 5.2 and 6.0 text files**

Release 6.0 programs including SEEHAR writes 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.

### 7.2.3 GEMPACK Documentation for Releases 6.0 and 6.0-001


Harrison, W.J. and K.R. Pearson (1998), *An Introduction to GEMPACK*, GEMPACK Document No. 1 **[GPD-1],** Monash University, Clayton, Fourth edition, October 1998.

Harrison, W.J. and K.R. Pearson (1994), *User's Guide to TABLO, GEMSIM and TABLO-generated Programs*, GEMPACK Document No. 2 **[GPD-2],** Monash University, Clayton, Second edition, April 1994.

Harrison, W.J. and K.R. Pearson (1993), *How to Create and Modify GEMPACK Header Array Files Using the Program MODHAR*, GEMPACK Document No. 3 **[GPD-3],** Monash University, Clayton, Third edition, April 1993.

Harrison, W.J. and K.R. Pearson (1998), *Release 6.0 of GEMPACK - New Features and Changes From Release 5.1 and 5.2*, GEMPACK Document No. 4 **[GPD-4],** Monash University, Clayton, Second edition, October 1998.

Harrison, W.J. and K.R. Pearson (1998 and 1999), *Installing and Using the Source-Code Version of GEMPACK on DOS/Windows PCs with Lahey Fortran*, GEMPACK Document No. 6 **[GPD-6],** Monash University, Clayton, Eighth edition, October 1998 (Release 6.0) and Ninth .edition, April 1999 (Release 6.0-001).

Harrison, W.J. and K.R. Pearson (1998 and 1999), *Installing and Using the Executable-Image Version of GEMPACK on DOS/Windows PCs*, GEMPACK Document No. 7 **[GPD-7],** Monash University, Clayton, Fifth edition, October 1998 (Release 6.0) and Sixth edition, May 1999 .(Release 6.0-001).

Harrison, W.J. and K.R. Pearson (1998), *Getting Started with GEMPACK: Hands-on Examples*, GEMPACK Document No. 8 **[GPD-8],** Monash University, Clayton, First edition, October 1998.

## *7.3 Release 5.2 (September 1996)*

### 7.3.1 New Features in Release 5.2

The new features and enhancements of Release 5.2 (September 1996) over and above Release 5.1 (April 1994) related mainly to the following areas.[132]

(1) When running GEMSIM or TABLO-generated programs, **extra TABLO-like statements** (for example, declaring extra sets or asking for extra WRITEs or DISPLAYs) can be included in the Command file. Such statements are carried out as if they were appended to the end of the original TABLO Input file. [See section 6.6 of GPD-3.]

(2) Arrays on Header Array files can now contain **set and element labelling information**, namely the names of the coefficients usually associated with the arrays, the names of the sets over which the arguments of the array run and the names of the elements of the sets involved with the array. TABLO-generated programs and GEMSIM now automatically write this information to any arrays they write to a Header Array file. This information is shown in SEEHAR output and also when SLTOHT produces Header Array output (the default) or when the new option 'SSE' is selected. It is also shown by VIEWHAR (see below). [See chapter 5 of GPD-4.]

(3) When GEMPACK runs on DOS machines under Microsoft Windows or Windows95, it can be run under a **new GEMPACK Windows interface** called **WinGEM**. Users can carry out modelling tasks using GEMPACK windows. We hope that this will make modelling easier for both new and experienced modellers using these machines. As part of this, a new Windows program VIEWHAR (written by our colleague Mark Horridge) allows windows-type access to examine the data in Header Array files. [You can think of VIEWHAR as a Windows version of SEEHAR.] [See section 2.1 of GPD-4.]

(4) **SET complement** syntax, for example

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

is allowed in TABLO Input files. [See section 4.6.4 of GPD-2.]

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

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

which says that the set SPCOM consists of all commodities c in COM for which TOTX(c) is greater than one-fifth of TOTY(c). [See section 4.6.5 of GPD-2.]

(6) Explicit **mappings between sets** (for example, a mapping from unique-product industries to unique-product commodities, mapping each industry to the commodity it makes) are available in TABLO Input files. [See section 4.8 of GPD-2.]

(7) A new **function $POS** is allowed which can be used to indicate the position number of an index or element in a set in TABLO Input files. [See section 4.4.4 of GPD-2.]

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

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

[See section 3.14 of GPD-2 and section 6.3 of GPD-3.]

(9) New **conditionals involving indices and/or set mappings** explicitly, for example "IF( i EQ j, <expression> )", are available in TABLO Input files. [See section 4.4.6 of GPD-2.]

---

[132] The text here is taken from the Release 5.2 documentation. In particular, "now" in this section means September 1996, when Release 5.2 was made available. However, cross references relate to the current (Release 8.0) versions of the GEMPACK documents.

(10) It is now possible to **write updated** (that is, post-simulation) **values of coefficients initialised via a FORMULA(INITIAL)**. [See section 4.11.8 of GPD-2.]

(11) New **sparse linear equation solving routines MA48** (sometimes a faster alternative to the MA28 routines) are now the default. [See section 12.1 of GPD-3.]

(12) You **can now specify in advance what accuracy you require** for simulations carried out via TABLO-generated programs or GEMSIM. You specify what accuracy you require (how many significant figures) and what percentage of results you require to be this accurate and the program works out how many steps and how many subintervals are required to achieve this accuracy. [See section 7.4 of GPD-3.]

(13) **Newton's method** for solving levels equations is now available. TABLO is used to add the appropriate Newton correction term to levels equations. When running a simulation using GEMSIM or the TABLO-generated program, the shock to the Newton variable can be added to the usual shocks in an Euler solution, or the Newton shock can be applied alone in a series of Newton correction steps applied after the original Euler multi-step simulation. [See section 7.5 of GPD-3.]

(14) Procedures for doing **systematic sensitivity analysis** and for handling inequalities including modelling of explicit **import and export volume quotas** in conjunction with any model solved using GEMPACK have been developed and automated recently. Details about these procedures are given in chapter 16 of the Release 7.0 version of GPD-3. **The procedures for handling quotas referred to here are now superseded by the Complementarity methods described in chapter 16 of the Release 8.0 version of GPD-3.**

(15) A facility for **translating GEMPACK Header Array files to GAMS** format. [There is also a complementary facility available in GAMS for translating GAMS data files to GEMPACK format.] See chapter 16 of GPD-4.

### 7.3.2 GEMPACK Documentation for Release 5.2


Harrison, W.J. and K.R. Pearson (1996), *An Introduction to GEMPACK*, GEMPACK Document No. 1 **[GPD-1]**, Third edition, September 1996, pp.250+15.

Harrison, W.J. and K.R. Pearson (1994), *User's Guide to TABLO, GEMSIM and TABLO-generated Programs*, GEMPACK Document No. 2 **[GPD-2]**, Second edition, April 1994, pp.138+14.

Harrison, W.J. and K.R. Pearson (1993), *How to Create and Modify GEMPACK Header Array Files Using the Program MODHAR*, GEMPACK Document No. 3 **[GPD-3]**, Third edition, April 1993, pp.27+4.

Harrison, W.J. and K.R. Pearson (1996), *Release 5.2 of GEMPACK - New Features and Changes From Release 5.1*, GEMPACK Document No. 4, First edition, September 1996, pp.98+9.

Harrison, W.J. and K.R. Pearson (1996), *Installing and Using GEMPACK on Macintosh PCs with MacFortran II*, GEMPACK Document No. 5B **[GPD-5B]**, Second edition, November 1996.

Harrison, W.J. and K.R. Pearson (1996), *Installing and Using GEMPACK on DOS, Windows or Windows 95 PCs with Lahey Fortran*, GEMPACK Document No. 6 **[GPD-6],** Fifth edition, September 1996, pp.23+5 (Release 5.2).

Harrison, W.J. and K.R. Pearson (1997), *Installing and Using GEMPACK on DOS/Windows PCs with Lahey Fortran*, GEMPACK Document No. 6 **[GPD-6],** Sixth edition, January 1997 (Release 5.2).

Harrison, W.J. and K.R. Pearson (1997), *Installing and Using the Source-code Version of GEMPACK on DOS/Windows PCs with Lahey Fortran*, GEMPACK Document No. 6 **[GPD-6],** Seventh edition, August 1997 (Release 5.2-002).

Harrison, W.J. and K.R. Pearson (1996), *Installing and Using the Executable-Image Version of GEMPACK on 80386/80486/Pentium PCs*, GEMPACK Document No. 7 **[GPD-7]**, Second edition, September 1996 (Release 5.2).

Harrison, W.J. and K.R. Pearson (1997), *Installing and Using the Executable-Image Version of GEMPACK on DOS/Windows PCs*, GEMPACK Document No. 7 **[GPD-7]**, Third edition, April 1997 (Release 5.2) and Fourth edition, September 1997 (Release 5.2-002).

## *7.4 Earlier Releases*

These were

Release 5.1 (April 1994)

Release 5.0 (April 1993)

Release 4.2.02 (April 1991)

Details can be found in the earlier versions of the GEMPACK documentation.

# 8. REFERENCES


Atkinson, Kendall E. (1989), *An Introduction to Numerical Analysis*, second edition, Wiley, New York.

Brooke, T., D. Kendrick and A. Meeraus (1988), *GAMS: A User's Guide*, The Scientific Press, Redwood City, California.

Codsi, G., K.R. Pearson and P.J. Wilcoxen (1992), 'General-Purpose Software for Intertemporal Economic Models', *Computer Science in Economics and Management* vol.5, pp.57-79. [A preliminary version was *Impact Preliminary Working Paper* No. IP-51, Melbourne (May 1991), pp.39.]

Codsi, G. and K.R. Pearson (1988), 'GEMPACK: General-Purpose Software for Applied General Equilibrium and Other Economic Modellers', *Computer Science in Economics and Management* vol.1, pp.189-207. [A preliminary version was *Impact Preliminary Working Paper* No. IP-39, Melbourne (July 1988), pp.31.]

Dixon P.B., B.R. Parmenter, J. Sutton and D.P. Vincent (1982), *ORANI: A Multisectoral Model of the Australian Economy*, North-Holland, Amsterdam.

Dixon, P.B., B.R. Parmenter, A.A. Powell and P.J. Wilcoxen [**DPPW**] (1992), *Notes and Problems in Applied General Equilibrium Economics*, North-Holland, Amsterdam.

Dixon, P.B. and M.T. Rimmer (2001), 'Dynamic, General Equilibrium Modelling for Forecasting and Policy: a Practical Guide and Documentation of MONASH', (Draft, May 2001), Centre of Policy Studies and Impact Project, Monash University. Selected chapters are available from *http://www.monash.edu.au/policy/monmod.htm*

Dixon, P.B. and M.T. Rimmer (2002), *Dynamic General Equilibrium Modelling for Forecasting and Policy: a Practical Guide and Documentation of MONASH*, North-Holland, Amsterdam.

Duff I.S. (1977), 'MA28 - A Set of FORTRAN Subroutines for Sparse Unsymmetric Linear Equations', Harwell Report R.8730 (HMSO, London), pp.104.

Harrison, W.J., Mark Horridge, K.R. Pearson and Glyn Wittwer (2002), 'A Practical Method for Explicitly Modeling Quotas and Other Complementarities',*Centre of Policy Studies and the Impact Project Preliminary Working Paper* No. IP-78, Melbourne (April), pp.19.

Harrison, W.J. and K.R. Pearson (1996), 'Computing Solutions for Large General Equilibrium Models Using GEMPACK', *Computational Economics*, vol. 9, pp.83-127. [A preliminary version was *Impact Preliminary Working Paper* No. IP-64, Monash University, Clayton (June 1994), pp.55.]

Harrison, W.J., K.R. Pearson and A.A. Powell (1996), 'Features of Multiregional and Intertemporal AGE Modelling with GEMPACK', *Computational Economics*, vol. 9, pp. 331-353. [A preliminary version was "Multiregional and intertemporal AGE modelling via GEMPACK", *Impact Preliminary Working Paper* No. IP-66, September 1994, pp. 21.]

Harrison, W.J., K.R. Pearson, A.A. Powell and E.J. Small (1994), 'Solving Applied General Equilibrium Models Represented as a Mixture of Linearized and Levels Equations', *Computational Economics*, vol. 7, pp. 203-223.

[A preliminary version was *Impact Preliminary Working Paper* No. IP-61, Monash University, Clayton, September 1993, pp. 20.]

Harrison, W.J. and E.J. Small (1993), 'TABLO Input Files for the Stylized Johansen, Miniature. ORANI and ORANI-F Models', *Impact Computing Document* No. C11-01, Monash University, Clayton (September 1993), pp.29.

Hertel, T.W., J.M. Horridge and K.R. Pearson (1992), 'Mending the Family Tree: A Reconciliation of the Linearized and Levels Schools of AGE Modelling', *Economic Modelling*, vol.9, pp.385-407. [A preliminary version was *Impact Preliminary Working Paper* No. IP-54, Melbourne (June 1991), pp.45.]

Hertel, Thomas W. (ed) (1997), *Global Trade Analysis: Modeling and Applications*, Cambridge University Press.

Hertel, T.W. and M.E. Tsigas (1993), 'GTAP Model Documentation', Department of Agricultural Economics, Purdue University, July 1993, pp.32+26.

Horridge, J.M., B.R. Parmenter and K.R. Pearson (1993), 'ORANI-F: A General Equilibrium Model of the Australian Economy', *Economic and Financial Computing*, vol.3, pp.71-140.

Kohlhaas, M. and K.R. Pearson (2002), 'Introduction to GEMPACK for GAMS Users', *Centre of Policy Studies and the Impact Preliminary Working Paper* No. IP-79, Melbourne (April), pp.39.

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

Malakellis, Michael (2000), *Integrated Macro-Micro-Modelling under Rational Expectations: With an application to Tariff Reform in Australia*, Physica-Verlag.

Pearson, K.R. (1988), 'Automating the Computation of Solutions of Large Economic Models', *Economic Modelling*, vol.5, pp.385-395. [A preliminary version was *Impact Preliminary Working Paper* No. IP-27, Melbourne (March 1986), pp.28.]

Pearson K.R. (1991), 'Solving Nonlinear Economic Models Accurately via a Linear Representation', *Impact Preliminary Working Paper* No. IP-55, Melbourne (July), pp.39.

Pearson, K.R. (1992) *Simulation Software for Use with 'Notes and Problems in Applied General Equilibrium Economics*, North-Holland, Amsterdam.

Pearson, K.R., Thomas W. Hertel and J. Mark Horridge (2002), 'AnalyseGE: Software Assisting Modellers in the Analysis of their Results', October 2002. [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.

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

Wilcoxen, P.J. (1989), 'Intertemporal Optimization in General Equilibrium: A Practical Introduction', *Impact Preliminary Working Paper* No. IP-45, Melbourne (December), pp.170.

# 9. GEMPACK DOCUMENTS[133]


Harrison, W.J. and K.R. Pearson (2002), *An Introduction to GEMPACK*, GEMPACK Document No. 1 **[GPD-1],** Monash University, Clayton, Sixth edition, October 2002, pp.207+9.

Harrison, W.J. and K.R.Pearson (2002), *TABLO Reference*, GEMPACK Document No. 2 **[GPD-2]** Monash University, Clayton, Fourth edition, October 2002, pp.191+10.

Harrison, W.J. and K.R.Pearson (2002), *Simulation Reference: GEMSIM, TABLO-generated Programs and SAGEM*, GEMPACK Document No. 3 **[GPD-3]**, Monash University, Clayton, Second edition, October 2002, pp.262+12.

Harrison, W.J. and K.R.Pearson (2002), *Useful GEMPACK Programs*, GEMPACK Document No. 4 **[GPD-4]** Monash University, Clayton, Second edition, October 2002, pp.138+10.

Harrison, W.J. and K.R. Pearson (2002), *Installing and Using the Source-Code Version of GEMPACK on Windows PCs with Lahey Fortran*, GEMPACK Document No. 6 **[GPD-6],** Monash University, Clayton, 11th edition, October 2002, pp.39+7.

Harrison, W.J. and K.R. Pearson (2002), *Installing and Using the Executable-Image Version of GEMPACK on Windows PCs*, GEMPACK Document No. 7 **[GPD-7],** Monash University, Clayton, 8th edition, October 2002, pp.33+6.

Harrison, W.J. and K.R. Pearson (2002), *Getting Started with GEMPACK: Hands-on Examples*, GEMPACK Document No. 8 **[GPD-8],** Monash University, Clayton, Third edition, October 2002, pp.110+8.


[133] 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".

# 10. INDEX


## —<—

<cmf>
  Use in Command files, 2-55

## —A—

Abbreviations
  % and #, 5-167
Absorption, 7-190
ACCUM
  Documentation. *See* chapter 10 of GPD-4
Accuracy, 2-74
Accuracy Summary
  Overall, 1-9
Adams, Philip, 1-19
ADD_HOMOTOPY qualifier, 1-8
Address
  Contact, 1-3
AnalyseGE, 3-109, 7-179. *See* chapter 2 of GPD-4
  Complementarities, 1-10
  Conditions, 1-10
  Data manipulation TAB file, 1-10
  Functions, 1-10
  Incentive to backsolve, 3-127
  Integer coefficients, 1-10
  Set mappings, 1-10
AnalyseGE example
  Data-manipulation TAB file. *See* section 6.8.2.1 of GPD-3
  Simulation. *See* section 6.1 of GPD-8
Analysing simulation results, 3-109
ASCII file, 5-171
Auxiliary files, 2-80
Auxiliary Statement,Table files, 2-45
AVC file, 1-10

## —B—

Backsolved variables
  On Extrapolation Accuracy file, 3-124
  On Soluiton file, 3-124
Backsolving, 3-124, 3-127
  AnalyseGE, 3-127
Balance of data base
  Checking, 4-153
  Checking updated data, 4-153
Batch operation, 5-160, 5-162
Binary files
  Use in GEMPACK, 5-171
Bug
  Reporting, 5-169

## —C—

C matrix, 2-72
  Condensation, 3-119
Calibration, 3-113
Cartesian product of sets, 1-7
Case sensitive, 3-108, 5-157
  TABLO, 3-98
Case sensitive file names, 5-173
Centre of Policy Studies, 1-19
  Address, 1-3
Change UPDATE, 3-109
Change variables, 3-100
Changing closure, 2-63
Changing shocks, 2-63
Characters
  Chinese, 5-174
  Non-English, 5-174
Charter, 2-62
Check
  TABLO, 3-119
Checking balance, 4-153
Chinese characters, 5-174
Choosing variables interactively. *See* chapter 17 of GPD-4
Closure, 2-46, 2-49
  Invalid, 7-183
  Specifying. *See* chapter 5 of GPD-3
Closure not valid, 1-9
CMBHAR
  Documentation. *See* chapter 4 of GPD-4
CMF file. *See* Command file
-cmf option, 5-166, 7-189
CMPHAR
  Documentation. *See* chapter 4 of GPD-4
Code
  TABLO, 3-119
Coefficient
  TABLO, 3-99
Coefficient initialisation, 1-9
Command file, 2-46, 2-49, 2-79
  Correcting errors, 3-131
  Documentation. *See* GPD-3
  For data-manipulation TAB file, 4-147
  Length of lines, 1-9
  Statements. *See* GPD-3
  Syntax. *See* GPD-3
  Syntax errors, 3-131
Command file statements
  Summary. *See* chapter 18 of GPD-3
Command line options
  Available on Unix machines, 7-183
  Available under DOS, 5-166
  Available under Unix, 5-166
  -cmf, 5-166, 7-189
  -lic, 5-167, 7-183
  -log, 5-166, 7-189
  -lon, 5-166, 7-189
  -los, 5-166, 7-183
  -sti, 5-166, 7-189
Command prompt, 2-25
Comments, 5-159
  In Command files. *See* section 2.7 of GPD-3
  In GEMPACK text data files. *See* section 6.1 of GPD-4
  In TABLO Input files. *See* section 4.1 of GPD-2
  Mapping files, 7-188
  Text files, 4-139

Compile, 2-46
Compile and link, 2-39. *See* section 1.2 of GPD-2
  Command prompt, 2-46
  On Windows PCs. *See* section 6.3 of GPD-6
  WinGEM, 2-30
Complementarities, 1-7, 1-14. *See* chapter 16 of GPD-3
Complementarity statement
  AnalyseGE, 1-10
Component numbers. *See* section 5.3 of GPD-3
Computation time
  Extrapolation, 2-75
Condensation, 3-119
Condensation actions
  In TAB file, 1-7
  On TABLO Input file, 3-128
Condense
  TABLO, 3-119
CONLF. *See* chapter 15 of GPD-4
Consequential error
  In TABLO Input file, 3-131
Contact
  GEMPACK, 1-3
ConvHAR. *See* chapter 15 of GPD-4
Correcting errors
  In Command files, 3-131
  In TABLO Input files, 3-129
CSV file
  SLTOHT, 2-59
CSV format, 4-136, 4-148
Cumulative results, 2-70
Cumulatively-retained endogenous
  Backsolves, 3-124
Current directory, 5-158
  WinGEM, 5-158, 5-159
CVL file, 1-10

## —D—

DAGG program, 4-155
Data
  Is it balanced?, 4-153
  Modifying, 4-152
  Post-simulation, 2-52, 2-57
  Pre-simulation, 2-57
  Transferring to/from GAMS. *See* chapter 16 of GPD-4
  Updated, 2-52, 2-57
Data bases, 7-181
  Converting Header Array files, 7-181
Data file, 2-79, 4-135, 5-171
  Header Array, 4-135, 4-136
  Text, 4-136
Data manipulation
  TABLO, 4-146
Decomposing simulation results, 7-179
Default response, 5-158
Demonstration Version, 1-15, 1-16
  File names, 5-174
DEVIA
  Documentation. *See* chapter 10 of GPD-4
Different machines, 1-1
Directory
  Invalid, 2-25
  No spaces, 2-25
Display
  TABLO, 3-99
Dixon, Peter, 1-19
Documentation, 6-177
DPPW
  Reference, 2-23

## —E—

Element names in TAB file, 1-7
Eliminated variables, 3-124
Empty sets, 7-188
Endogenous, 2-21
Environment file, 2-81, 5-172
Equation block, 2-72
Equations, 2-72
  Linearized, 2-21
  TABLO, 3-92, 3-99
Equations file, 2-79
  Connection to Equations Matrix, 2-72
  Contents, 2-72
  Production and use, 5-172
  Reason for the name, 2-72
Equations Matrix, 2-72
  Condensation, 3-119
Error trapping
  Invalid arithmetic, 7-182
  Logarithms, 7-182
  Overflow, 7-182
  Powers, 7-182
Euler's method, 2-74
Examples
  Hands-on. *See* GPD-8
Exclamation mark, 5-159
Executable image, 2-39
  Of a TABLO-generated program. *See* section 1.2 of GPD-2
Executable-image GEMPACK
  Installing on Windows PCs. *See* GPD-7
Executable-image licence limits, 1-16
Executable-image licences, 1-16
  Limited, 1-16
  Unlimited, 1-16
Executable-image Version, 1-15, 1-16
  File names, 5-174
  Limited, 1-16
  Unlimited, 1-16
Exogenous, 2-21
Extrapolation, 2-75
Extrapolation Accuracy file, 2-56, 5-172. *See* section 7.2 of GPD-3
  Contents, 2-75
Extrapolation Accuracy Summary, 2-76. *See* section 7.2 of GPD-3
  Number of accurate figures, 2-76

## —F—

F77L3 compiler, 5-168
  No longer supported, 1-7
FAQs, 1-3
FC5
  TABLO Code option, 1-8
File names, 5-173

Allowed, 5-173
Case sensitive ?, 5-173
Demonstration Version, 5-174
Depend on compiler, 5-173
Depend on operating system, 5-173
Executable-image Version, 5-174
Invalid, 2-25
LF90 and LF95 programs, 5-174
Long, 2-25, 5-174
No trailing spaces, 5-173
File names including spaces, 1-11
Files
Logical name, 3-99
Overview, 2-79
Suffixes, 2-79
TABLO, 3-99
Final_level statement, 1-9
FORMULA & EQUATION, 3-99
Fortran 77
Support may be phased out, 7-181
Fortran 77 compilers
No longer supported, 1-7
Fortran 90
Memory management, 5-168
Fortran compilers, 1-7, 1-15
F77L3, 5-168
Fortran 77, 5-168
Fortran 90, 5-168
LF90, 5-168
LF95, 5-168
Unix, 5-168
Windows PCs, 5-168
Frequently Asked Questions, 1-3
FTP, 4-137
Fujitsu Header Array files. *See* chapter 15 of GPD-4
Functions
CUMNORMAL, 7-184
NORMAL, 7-184
RANDOM, 7-184


## —G—


GAMS data files. *See* chapter 16 of GPD-4
GEMPACK
FAQs, 1-3
Frequently Asked Questions, 1-3
Hands-on introduction. *See* GPD-8
Versions, 1-15
World Wide Web, 1-3
GEMPACK documents, 6-177
GEMPACK Examples
Associated files. *See* chapter 1 of GPD-8
GEMPACK files
Overview, 2-79
Suffixes, 2-79
GEMPACK licence, 1-15, 5-167
Executable-image licence limits, 1-16
GEMPACK programs, 1-12
Common features, 5-157
Overview, 2-77
GEMPACK text data files
Documentation. *See* chapter 6 of GPD-4
GEMPACK-L
Mailing List, 1-3
GEMPIE
Documentation. *See* chapter 7 of GPD-4
Example, 2-47, 2-49
Print file, 2-39, 2-40, 2-47
GEMSIM
Documentation. *See* GPD-3
How to run, 2-49
GEMSIM Auxiliary files
Meaning, 2-40
GEMSIM Statement, Table files, 2-48
Gloss
TABmate, 3-93
Gragg's method, 2-74
Graphs
In reports, 2-59
Graphs of results, 2-62
GTAP, 1-19
GTAPVIEW TAB file, 4-154


## —H—


Hands-on examples. *See* GPD-8
HAR file. *See* Header Array file
Harrison, Paul, 1-19
Header Array files, 4-136
Creating, 4-135
Introduction, 4-136. *See* section 3.1 of GPD-4
Modifying, 4-135
Modifying data, 4-152
Headers, 4-136
Reserved, 4-136
Help
Online, 5-165
Hertel, Tom, 1-19
Homogeneity
Nominal. *See* chapter 13 of GPD-4
Real. *See* chapter 13 of GPD-4
Homogeneity simulations. *See* chapter 13 of GPD-4
Homotopy terms
Adding, 1-8
Horridge, Mark, 1-19, 4-155
How much data information, 4-138


## —I—


Impact Project, 1-3
Implementation, 2-21
Individual column results, 2-68
INF file. *See* Information file
Information file, 2-45, 2-48
Input redirection via "<"
Not recommended, 5-166
Installing Executable-image GEMPACK
On Windows PCs. *See* GPD-7
Installing Source-code GEMPACK
On Windows PCs. *See* GPD-6
Interactive operation, 5-160, 5-162
Internal program error, 5-169
Intertemporal models, 1-14. *See* chapter 7 of GPD-2
Introductory licence, 1-17
When required, 1-17
Invalid closure, 7-183
Invalid input
ASI, SIF, BAT, 5-162

## —J—

Johansen simulation, 2-65, 2-73
Johansen solution
  Meaning, 2-73

## —K—

Keyword, 3-92, 3-98
Kohlhaas, Michael, 1-7, 1-19

## —L—

Labelling information, 3-98
Lahey Header Array files. *See* chapter 15 of GPD-4
Large models
  Condensation, 3-119
Large-simulations licence, 1-18
  When required, 1-18
Left Hand Side Matrix. *See* LHS Matrix
Levels equations, 2-22, 3-87
Levels names during Condensation, 1-8, 3-119, 3-121, 3-125, 3-127, 3-128
Levels names in Command files, 1-8
Levels variables, 2-22, 3-100
LF90, 5-168, 7-181
LF95, 5-168, 7-181
LF95 Express, 7-181
LHS Matrix, 2-73
LHS Matrix singular, 1-9
Licence
  Executable-image licence limits, 1-16
  Introductory, 1-17
  Large-simulations, 1-18
Licence file, 1-18, 5-167
Licences for GEMPACK, 1-15
Limited Executable-image Version, 1-16
Limited Executable-image Version licences, 1-16
Line length
  In Command files. *See* section 2.7 of GPD-3
  In GEMPACK text data files. *See* section 6.2 of GPD-4
  In TABLO Input files. *See* section 4.1 of GPD-2
LINEAR_NAME qualifier, 1-8
LINEAR_VAR qualifier, 1-8
Linearized equations, 2-72, 3-87
-log
  Abbreviating filename, 5-167
Log file, 2-56, 2-81, 5-160
  Error, 5-164
  Terminal output, 5-163
Log file name
  Abbreviating on Command line, 1-11
-log option, 5-166, 7-189
Logarithms
  Error trapping, 7-182
-lon option, 5-166, 7-189
Long file names, 1-11, 2-25, 5-174, 7-181
  Do not use "%"or "#", 5-174
  Only use "simple" characters, 5-174
Long name
  Header Array file, 4-136
  On updated Header Array file, 1-9
LTG
  Compile and link, 2-39, 2-46
  For compiling and linking. *See* section 1.2 of GPD-2

## —M—

Macro variable
  Meaning, 2-24
Mailing List
  GEMPACK-L, 1-3
Malakellis, Michael, 1-14
Mapping
  Projection, 1-8
Mapping file, 2-81
Matrix
  Sparse, 2-74
MAXS operator in TAB file, 1-7
McDougall, Robert, 1-19
Memory allocation
  Fortran 90, 5-168
Midpoint method, 2-75
MINS operator in TAB file, 1-7
MKHAR
  Documentation. *See* chapter 11 of GPD-4
MKSOL
  Documentation. *See* chapter 11 of GPD-4
Model examples, 1-14
Model representation
  Levels,Mixed,Linear, 3-118
MODHAR, 4-136, 4-143
  Documentation. *See* chapter 3 of GPD-4
Modifying data, 4-152
Multi-step solution, 2-72, 2-74

## —N—

New features
  Release 5.2, 7-194
  Release 6.0, 7-185
  Release 6.0-001, 7-190
  Release 7.0, 7-179
  Release 8.0, 1-7
Nominal homogeneity. *See* chapter 13 of GPD-4
Non-linear equation, 2-74
Normal distribution, 7-184

## —O—

Omitting variables, 3-127
Online Help, 5-165
Options
  GEMPACK - BAT, 5-162
  GEMPACK - BPR, 5-164
  GEMPACK - DRE, 5-164
  GEMPACK - DRO, 5-164
  GEMPACK - LOG, 5-161, 5-163
  GEMPACK - SIF, ASI, 5-161
  GEMPACK Basic, 5-160
  TABLO - ASB, 3-126, 3-127
  TABLO Code - FC5, 1-8
  TABLO Code - PGS, 2-40
  TABLO Code - WFP, 2-39
Options menu
  GEMPACK programs, 5-160
Options SSS and SES for SEEHAR, 1-10

ORANIG
Checking balance of data, 4-154
ORANI-INT model, 1-14
Overall Accuracy Summary, 1-9
Overflow, 7-182

## —P—

Parameters
In equations, 3-88
Percentage change, 2-22
Percentage change variables, 3-100
Post-simulation data, 2-52, 2-57
Powell, Alan, 1-19
Powers
Error trapping, 7-182
Pre-simulation data, 2-52, 2-57
Print file
Meaning, 5-171
PROD operator in TAB file, 1-7
Product of sets, 1-7
Program error, 5-169
Program options. *See* Options
Programs
Common features, 5-157
Overview, 2-77
Projection mapping, 1-8
Prompts
For file names, 5-158

## —Q—

Quantifier, 3-92
Quantifier declaration, 3-98
Quotas, 1-7, 1-14. *See* chapter 16 of GPD-3

## —R—

Random numbers, 7-184
Range checking, 7-182
READ
TABLO, 3-99
Reading data, 4-135, 5-171
Real homogeneity. *See* chapter 13 of GPD-4
Reducing model size
Condensation, 3-119
Relative file names, 5-159
Reporting program errors, 5-169
Reports
Tables and graphs, 2-59
Rimmer, Maureen, 1-19
ROUND function, 1-7
RunDynam, 7-180. *See* chapter 2 of GPD-4
Demonstration version, 1-10
RunGDYN, 7-180
RunGEM. *See* chapter 2 of GPD-4
RunMONASH, 7-180
Demonstration version, 1-10
RWHAR
Documentation. *See* chapter 11 of GPD-4
RWSOL
Documentation. *See* chapter 11 of GPD-4

## —S—

SAGEM, 2-65
Advantages and disadvantages, 2-71
Documentation. *See* chapter 10 of GPD-3
No updated data, 2-70
Scalar variable
Meaning, 2-24
SEEHAR, 4-136, 4-148, 7-181
Documentation. *See* chapter 4 of GPD-4
SEENV
Documentation. *See* chapter 12 of GPD-4
Sensitivity analysis. *See* Systematic Sensitivity Analysis
Set
Empty, 7-188
TABLO, 3-98
Set and element labelling, 4-141
Documentation. *See* chapter 5 of GPD-4. *See* chapter 5 of GPD-4
Set equality, 1-7
Set mappings, 7-180
Set product, 1-7
Shock file, 2-80
Shocks
Change, 1-9
From Header Array files, 7-180
Percentage change, 1-9
Specifying. *See* chapter 5 of GPD-3
Target, 7-184
Simulation
Meaning, 2-21
Several at once, 2-65
Steps, 2-37
Stylized Johansen example, 2-23
Simulation results
Analysing, 3-109
SLC file, 1-10, 2-80, 7-180
SLTOHT, 4-148
CSV file, 2-59
Documentation. *See* chapters 8 and 9 of GPD-4
Solution
Computation time, 2-75
Solution files, 2-39, 2-40, 2-46, 2-49, 2-79
Documentation. *See* chapter 8 of GPD-3
Solution methods
Documentation. *See* chapter 7 of GPD-3
Solving models, 2-21
Source-code GEMPACK
Installing on Windows PCs. *See* GPD-6
Source-code versions, 1-15
Spaces in file names, 7-181
Spreadsheet, 4-136
CSV file, 2-59
Separator, 4-148
SLTOHT, 2-59
Spreadsheet output from SEEHAR
Options SSS and SES, 1-10
SSA. *See* Systematic Sensitivity Analysis. *See* chapter 2 of GPD-4
Statements in TABLO Input file, 3-98
STI file. *See* Stored-input file
-sti option, 5-166, 7-189
Stored-input files, 2-81, 5-160
Creating via "sif" option, 4-143

Example, 4-143
Meaning, 5-160
On command line, 5-161
Re-using, 4-144
Terminal output, 5-163
Via option "sif", 4-143
Structurally singular model, 1-9
Stylized Johansen model
Checking balance of data, 4-153
Checking balance of updated data, 4-153
Data, 3-90
Description, 2-23
Equations, 3-87
Equations Matrix C, 2-72
GEMSIM simulation, 2-49
SAGEM simulation, 2-67
TABLO Input file, 3-95
TABLO-generated simulation, 2-46
Substituting variables
AnalyseGE, 3-127
Substitution, 3-119, 3-127
Subtotal, 2-70, 7-179
Documentation. *See* chapter 11 of GPD-3
In DEVIA and ACCUM, 1-10
SUM
TABLO, 3-99
SUMEQ
Documentation. *See* section 13.2 of GPD-4
SUMHAR, 4-136
Documentation. *See* chapter 4 of GPD-4
Systematic Sensitivity Analysis, 7-180. *See* chapter 2 of GPD-4
System-initiated Coefficients
On SLC,CVL,UDC,AVC files, 1-10


## —T—


TAB characters
In Command files, 1-11
In Stored-input files, 1-11
TAB file. *See* TABLO Input file
Interesting use of sets, 4-154
Tableau
Equations Matrix C, 2-72
Tables
In reports, 2-59
TABLO
How to run, 2-45, 2-48
TABLO Code option FC5, 1-8
TABLO Input file, 2-21, 2-79, 3-85, 3-91
Comment, 3-98
Consequential error, 3-131
Correcting errors, 3-129
Data manipulation, 4-146
Levels,Mixed,Linear, 3-118
Statements, 3-98
TABLO language, 3-101
TABLO stages, 3-119
TABLO syntax
Example, 3-92
TABLO-generated programs
Auxiliary files, 1-9
Compile and link on Windows PCs. *See* section 6.3 of GPD-6
Compiling and linking. *See* section 1.2 of GPD-2
Documentation. *See* GPD-3
Executable image, 2-80. *See* section 1.2 of GPD-2
Fortran file, 2-80
Meaning, 2-39
TABmate, 7-182. *See* chapter 2 of GPD-4
Gloss, 3-93
Text editor, 3-91
Target shock, 7-184
Tariff-rate quotas, 1-14. *See* chapter 16 of GPD-3
Terminal input
Comments, 5-159
Terminal output
Suppressing, 5-163
Text data files, 4-136
Documentation. *See* chapter 6 of GPD-4
Text editor
TABmate, 3-91
Text files, 4-136
Input to MODHAR, 4-139
Use in GEMPACK, 5-171
TEXTBI
Documentation. *See* chapter 14 of GPD-4
TG program. *See* TABLO-generated program
Trailing spaces in file names
Not allowed, 5-173
Transfer statements, 7-183
Transferring data
Between PCs and Unix machines, 4-137
Transferring data to/from GAMS. *See* chapter 16 of GPD-4
TRUNC0 function, 1-7
TRUNCB function, 1-7


## —U—


UDC file, 1-10
Unix/Command prompt, 2-25
Unlimited Executable-image Version, 1-16
Unlimited Executable-image Version licences, 1-16
Update
Example, 3-109
Meaning, 3-109
Product, 3-109
Purpose, 3-102, 3-107
TABLO, 3-102, 3-107
UPDATE(CHANGE), 3-109
Updated data, 2-52, 2-57
Is it balanced?, 4-153
Not produced by SAGEM, 2-70
Updated data file, 5-172
User input
Condensation in TABLO, 3-121
GEMPIE Example, 2-47, 2-49
MODHAR Modify data, 4-144
SEEHAR, 2-43
TABLO - Step 1, 2-45


## —V—


Variables, 2-72
Change, 3-100
Levels, 2-22
Percentage change, 2-22, 3-100

TABLO, 3-98
Vector variable, 2-72
Meaning, 2-24
Verbal description, 2-56
Versions of GEMPACK, 1-15
ViewHAR, 4-136, 7-182. *See* chapter 2 of GPD-4
Adding labelling information, 4-151
Creating new header, 4-149
Creating new Header Array file, 4-149
Modifying data, 4-150
Pasting into from spreadsheet, 4-152
ViewSOL, 7-182. *See* chapter 2 of GPD-4
Export, 2-59
Two solutions at once, 3-123

## —W—

Wilcoxen, Peter, 1-19
Windows 3.1
No longer supported, 7-182
WinGEM. *See* chapter 2 of GPD-4
Current directory, 5-158, 5-159
Work files, 2-81
Working directory. *See* Current directory
World Wide Web, 1-3
WRITE
TABLO, 3-99

## —X—

XTransfer statements, 7-183