GEMPACK USER DOCUMENTATION

Release 7.0

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

Fifth edition
October 2000

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

Fifth edition
October 2000

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/80486 PCs running Windows 95, 98, NT or 2000,
- Unix machines, and
- other mainframe, mini and microcomputers with an ANSI standard Fortran 77 or 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 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)

Table of Contents

1. INTRODUCTION 1-1

1.1 The Programs 1-2

1.2 Different Versions of GEMPACK and Associated Licences 1-3
1.2.1 Source-code Versions and Licences 1-3
1.2.2 Limited Executable-image Version and Licence 1-3
1.2.3 Unlimited Executable-image Version and Licence 1-4
1.2.4 Demonstration Version 1-4
1.2.5 Introductory Licence 1-5
1.2.6 Large-simulations Licence 1-5
1.2.7 WinGEM - The Windows Interface to GEMPACK 1-6
1.2.8 RunGEM - A Windows Program for Simulations with Different Models 1-6
1.2.9 Windows GEMPACK Programs Written by Mark Horridge 1-6
1.2.10 AnalyseGE - a New Way of Analysing Simulation Results 1-6

1.3 Models Supplied with GEMPACK 1-7

1.4 Documentation of GEMPACK 1-8

1.5 A Guide to This Document 1-8
1.5.1 For New GEMPACK Users 1-8
1.5.2 Changes for GEMPACK Release 7.0 From Earlier Versions of GEMPACK 1-9

1.6 Contacting the Centre of Policy Studies / Impact Project 1-15

1.7 GEMPACK World-Wide Web Site 1-15

1.8 GEMPACK-L Mailing List 1-15

1.9 Acknowledgments 1-16

2. HOW TO CARRY OUT SIMULATIONS WITH MODELS 2-17

2.1 The Steps in Carrying Out a Simulation 2-19
2.1.1 Steps using a TABLO-generated Program 2-20
2.1.2 Steps using GEMSIM 2-22

2.2 An Example Simulation with Stylized Johansen 2-24
2.2.1 Introduction to the Stylized Johansen Model 2-24
2.2.2 The Simulation 2-25

2.3 Installing GEMPACK Software 2-26

2.4 Using GEMPACK: WinGEM or Unix/Command Prompt 2-26

2.5 WinGEM : Stylized Johansen Example Simulation 2-27
2.5.1 Starting WinGEM 2-27
2.5.2 Preparing a Directory for Model SJ 2-27
2.5.3 Setting the Working Directory 2-28
2.5.4 TABLO-generated Program or GEMSIM? 2-28
2.5.5 The Example Simulation using a TABLO-generated Program 2-29
2.5.6 The Example Simulation using GEMSIM 2-32

2.5.7 GEMSIM or TABLO-generated Program? 2-34

2.6 Unix/Command prompt : Stylized Johansen Simulation 2-35
2.6.1 The Example Simulation using a TABLO-generated Program 2-35
2.6.2 The Example Simulation using GEMSIM 2-38
2.6.3 GEMSIM or TABLO-generated Program? 2-39

2.7 Interpreting the Results 2-40

2.8 Specifying a Simulation 2-43
2.8.1 Specifying a Simulation via a GEMPACK Command file 2-43

2.9 The Updated Data - Another Result of the Simulation 2-47

2.10 How to Carry out Several Simulations At Once 2-48
2.10.1 Preparing an Equations file for use by SAGEM 2-48
2.10.2 Running Several Johansen Simulations using SAGEM 2-49
2.10.3 The Results of These Johansen Simulations 2-51
2.10.4 The Command File SJLBJ.CMF for these Johansen Simulations 2-53
2.10.5 SAGEM - Advantages and Disadvantages 2-54

2.11 How Johansen and Multi-step Solutions are Calculated 2-55
2.11.1 The Linearized Equations of a Model 2-55
2.11.2 Johansen Solutions 2-56
2.11.3 Multi-step Simulations and Accurate Solutions of Nonlinear Equations 2-57

3. BUILDING OR MODIFYING MODELS 3-61

1.1 Writing Down the Equations of a Model 3-63
1.1.1 Writing Down the Equations of Stylized Johansen 3-63

1.2 Data Requirements for the Linearized Equations 3-66
1.2.1 Data Requirements for Stylized Johansen 3-66

1.3 Constructing the TABLO Input File for a Model 3-67
1.3.1 Constructing Part of the TABLO Input File for Stylized Johansen 3-67
1.3.2 The TABLO Input File for the Stylized Johansen Model 3-70
1.3.3 Completing the TABLO Input File for Stylized Johansen 3-73
1.3.4 Change or Percentage-Change Variables 3-75
1.3.5 TABLO Language - Syntax and Semantics 3-76

1.4 Constructing Data Files 3-77
1.4.1 Constructing the Data File for Stylized Johansen 3-77
1.4.2 Modifying Data Using MODHAR 3-81
1.4.3 Using TABLO for Data Manipulation 3-82
1.4.4 Header Array and Text Data Files 3-83

1.5 Linearized TABLO Input Files 3-86
1.5.1 A Linearized TABLO Input File for Stylized Johansen 3-87
1.5.2 Noteworthy Features in the Linearized TABLO Input File 3-90
1.5.3 Writing UPDATE Statements 3-93

1.6 Levels TABLO Input Files 3-95
1.6.1 Levels TABLO Input File for Stylized Johansen 3-95

1.7 Mixed, Linearized or Levels TABLO Input Files? 3-100

1.8 TABLO Linearizes Levels Equations Automatically 3-101

1.9 Condensing Models 3-102
1.9.1 Substituting Out Variables 3-102
1.9.2 Backsolving for Variables 3-104
1.9.3 Omitting Variables 3-106
1.9.4 Stopping and Restarting TABLO 3-106

1.10 Creating the TABLO Input File for Your Own Model 3-108

4. COMMON FEATURES OF GEMPACK PROGRAMS 4-109

4.1 Responding to Prompts (Upper case or lower case) 4-109
4.1.1 Default Response to Questions and Prompts 4-109

4.2 Comments in Input from the Terminal 4-110

4.3 Interactive and Batch Operation, Stored-input and Log Files 4-110
4.3.1 Invalid Input When Using Options 'sif' or 'asi' 4-112
4.3.2 Differences Between Batch and Interactive Program Dialogues 4-112
4.3.3 Terminal Output and Log Files 4-113
4.3.4 LOG Files and Early Errors 4-114

4.4 Other Program Options 4-114

4.5 Specifying Various Files on the Command Line 4-115
4.5.1 Command Line Option “-los” (alternative to “-lon”) 4-115
4.5.2 Command Line Option “-lic” to Specify the GEMPACK Licence 4-116

4.6 Fortran Compilers and Memory Management 4-116

4.7 Error Messages 4-117

4.8 GEMPACK File Types, Names and Suffixes 4-118
4.8.1 Files with System-determined Suffixes 4-118
4.8.2 Suffixes of Other Files 4-119
4.8.3 Files - Binary or Text? 4-119
4.8.4 Why So Many Files? 4-120

4.9 Allowed File and Directory Names 4-121
4.9.1 Are File Names Case Sensitive ? 4-121
4.9.2 File Names on Windows PCs – Programs Made with LF90 or LF95 4-121
4.9.3 File Names on Windows PCs – Programs Made with F77L3 4-122
4.9.4 File Names Containing Spaces 4-122

5. A GUIDE TO THE REST OF THE DOCUMENTATION 5-125

5.1 Other Relevant Publications 5-126

6. RECENT GEMPACK RELEASES 6-127

6.1 Release 6.0 (October 1998) and Release 6.0-001 (March 1999) 6-127
6.1.1 New Features in Release 6.0 (October 1998) 6-127
6.1.2 New Features in Relase 6.0-001 (March 1999) 6-132
6.1.3 GEMPACK Documentation for Releases 6.0 and 6.0-001 6-135

6.2 Release 5.2 (September 1996) **6-136**
6.2.1 New Features in Release 5.2 6-136
6.2.2 GEMPACK Documentation for Release 5.2 6-137

6.3 Earlier Releases **6-138**

7. REFERENCES **7-139**

8. GEMPACK DOCUMENTS **8-141**

9. INDEX **9-143**

CHAPTER 1

1. Introduction

This document is an introduction to the current version (Release 7.0, October 2000) 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 is 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/80486 PCs running Windows 95, 98, NT or 2000,
- Unix machines, and
- other mainframe, mini and microcomputers with an ANSI standard Fortran 77 or 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 5 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 8 gives the full list of all of the different GEMPACK documents and chapter 5 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 common features of the different GEMPACK programs.

Chapter 1 contains a summary of changes for the current Release 7.0 from earlier versions of GEMPACK. Chapter 6 contains information about recent GEMPACK releases before the current one.

1.1 The 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

2. Utility programs:

SUMEQ	for information about the numerical equations
SUMHAR,SEEHAR,CMPHAR,CMBHAR	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

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

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

4. Windows programs (which are only available for pentium/80486 PCs running Windows 95, 98, NT or 2000):

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

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.

1.2 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 1.2.1 below). The other versions of GEMPACK (Limited and Unlimited Executable-image and Demonstration Versions) are only available for pentium/80486 PCs running Windows 95, 98, NT or 2000.[1]. 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 7.0 of GEMPACK.

For more information, contact us at the Centre of Policy Studies / Impact Project (see section 1.6 below).

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

1.2.1 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/80486 PCs running Windows 95, 98, NT or 2000, and for Unix machines. Other machines may be added in the future.

1.2.2 Limited Executable-image Version and Licence

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 restrictions are on the size of the models that can be handled.[2]

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

[1] 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.

[2] 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.

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.

1.2.3 Unlimited Executable-image Version and Licence[3]

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.

1.2.4 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 1.3 below except for ORANI-G, ORANI-F 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 from address

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

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

1.2.5 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 is new with 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.[4] TABLO-generated programs produced with Release 6.0 (or later) of GEMPACK require an Introductory licence (or a Large-simulations licence – see section 1.2.6 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 1.2.2 above). Thus, for example, a modeller with the TABLO-generated program for the GTAP model will probably 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.

(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 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.

1.2.6 Large-simulations Licence

Large-simulation licences were introduced with Release 6.0 of GEMPACK to satisfy requirement (i) in section 1.2.5 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.

---

[4] 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.

If you have a Release 6.0 Large-simulations licence, it will still let you run Release 7.0 TABLO-generated programs.

1.2.7 WinGEM - The Windows Interface to GEMPACK

On pentium/80486 PCs running Microsoft Windows 95, 98, NT or 2000, the Windows interface **WinGEM** is available. This allows modellers to carry out most modelling tasks in a familiar Windows environment. WinGEM can be used with Source-code, Executable-image or the Demonstration Version of GEMPACK. It is available from the GEMPACK Web site (whose address is given in section 1.2). WinGEM can only be used in conjunction with Release 5.2 (or later) of GEMPACK: it cannot be used with Release 5.1 or earlier.

See section 2.1 of GPD-4 for more information about WinGEM. Chapter 2 of GPD-8 contains a hands-on introduction to GEMPACK using WinGEM.

1.2.8 RunGEM - A Windows Program for Simulations with Different Models

RunGEM is a Windows program[5] which makes it easy for users to carry out simulations with any model which has been implemented using GEMPACK. Users of RunGEM do not need any previous experience with GE modelling; in particular, they need have no prior knowledge of GEMPACK or Command files. RunGEM provides a windows environment for specifying the ingredients of a simulation (including the pre-simulation data file, the closure, the shocks and the names of the output files).

Whereas WinGEM is aimed at modellers (those who build or modify models), RunGEM is aimed at **model users** (those who use a model built by someone else to carry out simulations).

RunGEM makes it very 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 underlying parameters (for example, Armington parameters) of your model, or to changes in the shocks.

See section 2.5 of GPD-4 and chapter 5 of GPD-8 for more information about RunGEM.

1.2.9 Windows GEMPACK Programs Written by Mark Horridge

Other Windows programs **ViewHAR**, **ViewSOL** and **TABmate** are available within GEMPACK for use on PCs. These programs written by Mark Horridge, a colleague at the Centre of Policy Studies, have been improved continuously for several years. ViewHAR provides easy access to Header Array data files. ViewSOL is used for viewing GEMPACK Solution files. TABmate provides special help in editing TABLO Input files.

Each of these programs has its own on-line help. A brief introduction is given in sections 2.2, 2.3 and 2.4 of GPD-4.

1.2.10 AnalyseGE - a New Way of Analysing Simulation Results

AnalyseGE is a new windows program (written for Release 7.0 of GEMPACK) designed to assist modellers in the analysis of their simulation results.

---

[5] RunGEM was new for Release 6.0 of GEMPACK. Systematic sensitivity analysis was added later, for Release 6.0-001.

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

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

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

Further details on AnalyseGE are available in its own on-line help file and a brief introduction is given in section 2.6 of GPD-4.

1.3 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-F, the forecasting version of the ORANI model of the Australian economy, as documented in Horridge *et al* (1993),
- ORANI-G, a comparative-static version of the ORANI model of the Australian economy, similar to ORANI-F (see above) but without the forecasting capability,
- 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 three 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).

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

GEMPACK software makes it easy to transfer models between different computers (including different operating systems), as described in chapter 11 of GEMPACK document GPD-4, so you can easily exchange models with other modellers using GEMPACK.

1.4 Documentation of GEMPACK

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

For Release 7.0 of GEMPACK the documentation has been rewritten to incorporate developments for Release 5.2, Release 6.0 and Release 7.0 into the appropriate places in the documentation.

The GEMPACK documents have been rearranged so that the program TABLO is described in GPD-2 and information for running simulations, including GEMPACK Command file syntax, is given in GPD-3. All the utility programs including the Windows programs are described in GPD-4.

GPD-8 is a hands-on introduction to GEMPACK. 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 5. You should be familiar with this document GPD-1[6], 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

1.5 A Guide to This Document

We provide two alternative guides. Readers new to GEMPACK should read the one in section 1.5.1 while readers familiar with Release 6.0 (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.

1.5.1 For New GEMPACK Users

Chapter 2 tells you how to carry out simulations with existing models. We suggest that you read this in detail and (provided you have access to a machine with GEMPACK) carry out the simulations described there for yourself. See the sub-section headed "Getting Started" later in this section for suggestions as to how you can begin carrying out simulations.

---

[6] 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".

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.

When you want to know more about running the GEMPACK programs, read chapter 4. This chapter gives detailed suggestions for more efficient use of the programs whether you are running them interactively or in batch mode.

Chapter 5 is a guide to the rest of the GEMPACK documentation.

Getting Started

- If you are using a Windows PC (Windows 95, 98, NT or 2000), we suggest that you start working with WinGEM, the Windows version of GEMPACK. A hands-on introduction to GEMPACK using WinGEM is given in chapter 2 of GPD-8, which we encourage you to work through.
- We encourage users on other machines (including Unix machines) to work through chapter 3 of GPD-8 which contains suggestions for hands-on computing you can carry out to familiarise yourself with the different GEMPACK programs and the tasks they can help you with.

1.5.2 Changes for GEMPACK Release 7.0 From Earlier Versions of GEMPACK

This is addressed to readers who have used Release 6.0 or Release 5.2 of GEMPACK.

We believe that the most important new features in Release 7.0 are:

- 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.[7]

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 linearised 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.

[7] Descriptions of the new features in Releases 6.0 and 5.2 can be found in chapter 6.

See section 1.2.10 above and section 2.6 of GPD-4 for more details.

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 5.5.3 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 4.9 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 4.6 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.[8]

12. We are no longer supporting **Windows 3.1.**

    The Windows programs (including WinGEM, GemEdit, RunGEM, ViewHAR, ViewSOL,

[8] 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.

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 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 15.4 and 15.5 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 4.5.1 for more details.

The command line option **`-lic <licence_filename>`**
can be used to specify the name of the GEMPACK licence file. See section 4.5.2 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 4.5 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 section 3.9.1 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 5.5.5 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.

Various other changes were made for Release 6.0-001 (March 1999).

These are listed in section 6.1.2. We encourage users of Release 6.0 who have not used Release 6.0-001 or later to read these before beginning to use Release 7.0.

New features in Release 6.0 and Release 5.2.

These are listed in sections 6.1 and 6.2 respectively. We encourage users of earlier versions of GEMPACK who are not familiar with the changes to read these sections before beginning to use Release 7.0.

1.6 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
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

1.7 GEMPACK World-Wide Web Site

The GEMPACK World-Wide Web site is 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. 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 is also an alternative GEMPACK web site at address

http://www.gempack.com.au

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

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

1.8 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**

1.9 Acknowledgments

We are grateful to Alan Powell for his support during all the years that GEMPACK has taken to develop. His positive outlook, many 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) 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 all those, including James Giesecke, Kevin Hanslow, Tom Hertel, Robert McDougall, Allan Rae, Hans van Meijl, Frank van Tongeren, Terrie Walmsley, Ronald Wendner, Glyn Wittwer and our colleagues at the Centre of Policy Studies, who have beta tested Release 7.0. We are grateful to those who beta tested earlier releases.

We are grateful to Tom Hertel and his 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.

Most importantly, we are indebted to our colleagues at the Centre of Policy Studies at Monash University for their encouragement and support. They provide us with an ideal environment in which to develop this type of software.

CHAPTER 2

2. How to Carry Out Simulations with Models

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, it 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.]

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

Details about linearizing the equations, the syntax of the TABLO Input file and the preparation of data files are given in chapter 3. Here we show you how to carry out simulations with an existing model (that is, one built by someone else). In this discussion, we use as an example the Stylized Johansen model described in Chapter 3 of Dixon et al (1992), hereafter referred to as **DPPW**.

This 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 The Steps in Carrying Out a Simulation

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. The details of these steps are a little different depending on whether you have a Source-code or Executable-image version of GEMPACK.

Source-code version of GEMPACK

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 2.1.1, and these steps are illustrated in Figure 2.1.1.

Executable-image or Demonstration version of GEMPACK

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 2.1.2, and these steps are illustrated in Figure 2.1.2.

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

We illustrate these steps in detail, in the context of a specific simulation with the Stylized Johansen model, in sections 2.5 and 2.6 below.

2.1.1 Steps using a TABLO-generated Program

Step 1. Computer Implementation of the Model

Step 1(a).

Process the TABLO Input file for the model by running the program TABLO. At the Code stage, 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 of the model, produced in Step 1(a). This will produce an **executable image** of the TABLO-generated program.

Step 2. Simulation

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 are to be read and describes the closure (that is, the exogenous and endogenous variables) 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

Run the GEMPACK program GEMPIE to convert the solution produced in Step 2 to a **GEMPIE Print file**. This file can be printed (or edited).[9]

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. (Of course Steps 1(a) and 1(b) must be repeated if you change the TABLO Input file for the model in any way.)

[9] If you are working on a Windows PC, an alternative way of viewing the simulation results is to run the windows program ViewSOL. This alternative is described in GPD-8.

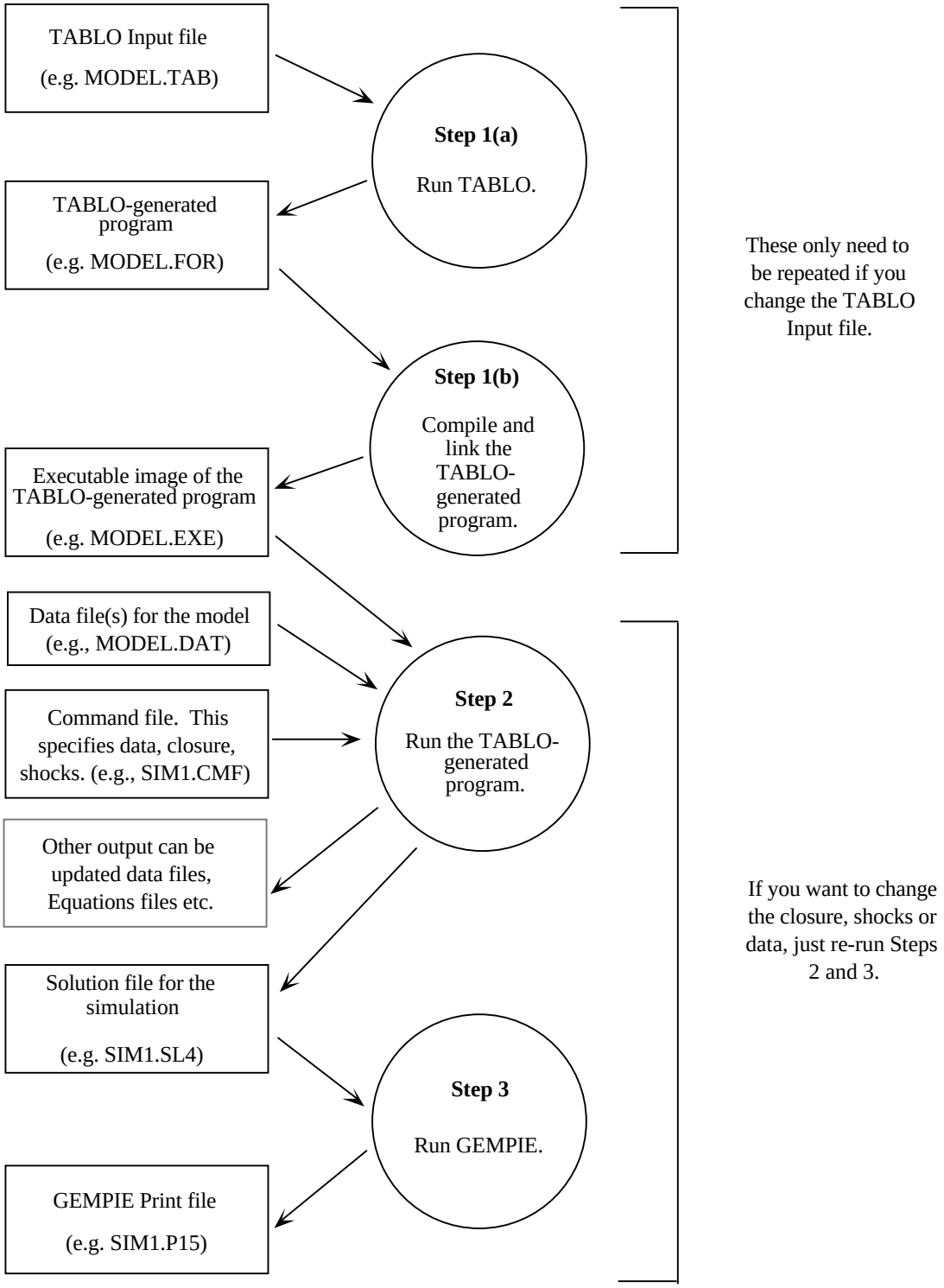


Figure 2.1.1 The Steps in Carrying Out a Simulation using a TABLO-generated Program

2.1.2 Steps using GEMSIM

This is an alternative method used for carrying out simulations which uses the program GEMSIM instead of writing a TABLO-generated program. These steps are illustrated in figure 2.1.2.

If you have a Source-code version of GEMPACK, and a suitable Fortran compiler, you can use either of these methods. If you have an **Executable-Image version** or the **Demonstration Version**, you must use the GEMSIM method described below.

Step 1. Computer Implementation of the Model

Process the TABLO Input file for the model by running the GEMPACK program TABLO. At the Code stage, 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 2.1.1 above is what initiates the GEMSIM route rather than the TABLO-generated program route.

Step 2. Simulation

Run the GEMPACK program GEMSIM [10] 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 are to be read and describes the closure (that is, the exogenous and endogenous variables) 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

Run the GEMPACK program GEMPIE to convert the solution produced in Step 2 to a **GEMPIE Print file**. This is a text file which can be printed (or edited).[11]

Note that Step 1 above is very similar to Step 1(a) in the TABLO-generated program case (see section 2.1.1 above). Step 1(b) in section 2.1.1 has no analogue in the GEMSIM case. Step 2 is different only in that GEMSIM is run rather than the TABLO-generated program. Step 3 is identical in the two cases.

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

[11] If you are working on a Windows PC, an alternative way of viewing the simulation results is to run the windows program ViewSOL. This alternative is described in GPD-8.

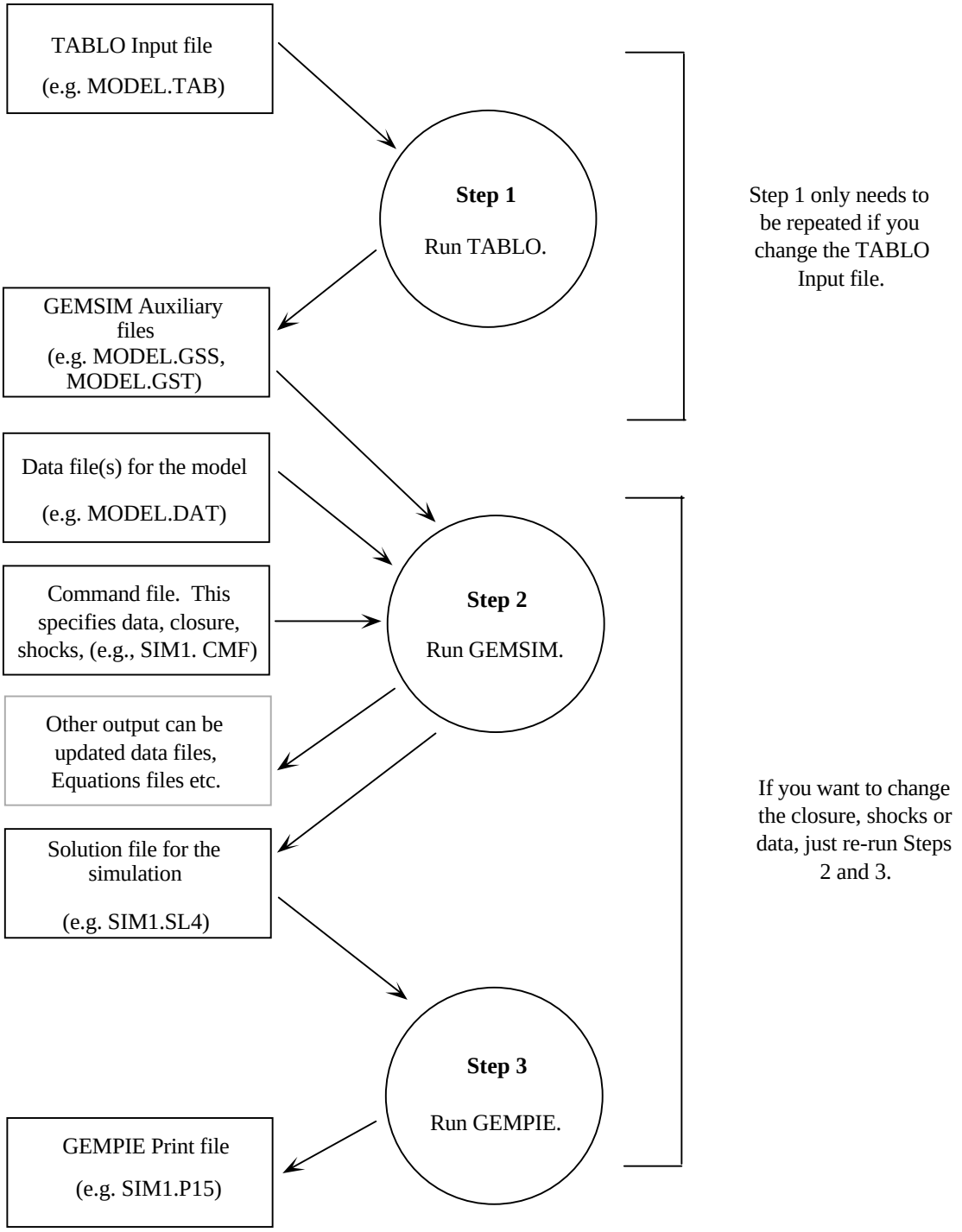


Figure 2.1.2 The Steps in Carrying Out a Simulation using GEMSIM

2.2 An Example Simulation with Stylized Johansen

2.2.1 Introduction to the Stylized Johansen Model

The 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 2.2.1a, which is the same as Table E3.3.1 in DPPW (i.e. 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 2.2.1a: Input-output Data Base for Stylized Johansen

In the GEMPACK implementation, the levels variables are as in Table 2.2.1b.

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 2.2.1b: 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** and to the others ('Y' is the only one here) 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 3.3.2. Full details about the model can be found in Chapter 3 of DPPW.

2.2.2 The Simulation

As an example of a simulation on the Stylized Johansen model, we will choose a closure with supplies of the two factors, labor and capital, as 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).

If you have access to a machine with GEMPACK installed, we suggest you actually carry out the steps as described below. Note that, when we refer to file names below, we are giving typical file names (such as on a Windows PC, for example). As indicated earlier, file names can vary from machine to machine.

2.3 Installing GEMPACK Software

If your computer is a PC, 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 (Windows 95, 98, NT or 2000), you should also install the Windows programs

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

2.4 Using GEMPACK: WinGEM or Unix/Command Prompt

There are two ways of operating GEMPACK.

- One method is working at the Command line under Unix, or on DEC VAX/VMS and Alpha/VMS machines. 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 2.5 about WinGEM and follow the instructions in section 2.6.

---

[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 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 4.9 for more details.

For those using **WinGEM,** follow the instructions in sections 2.5 to start WinGEM and copy the Stylized Johansen files to a working directory. Then carry out the examples running a simulation for the Stylized Johansen model.

2.5 WinGEM : Stylized Johansen Example Simulation

2.5.1 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.]

WinGEM - GEMPACK for Windows

File Simulation HAFiles OtherTasks Programs Options Window Help

2.5.2 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 Windows 95, 98, NT or 2000, 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**

2.5.3 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**).

2.5.4 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 2.5.5 below. Alternatively, you could also use the GEMSIM method (as spelled out in section 2.5.6 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.

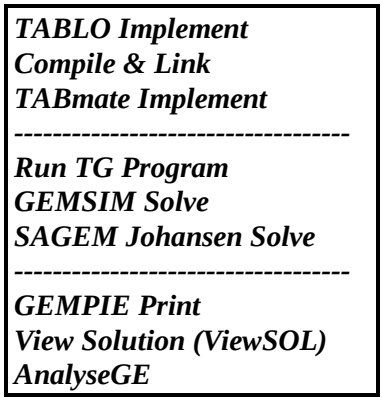
If you have an Executable-image version of GEMPACK, or the Demonstration version of GEMPACK, you must carry out simulations using the program GEMSIM; the steps in this case are spelled out in section 2.5.6 below.

In either case we illustrate the steps for the Stylized Johansen simulation described in section 2.2.2 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 2.5.5 and go straight to section 2.5.6.

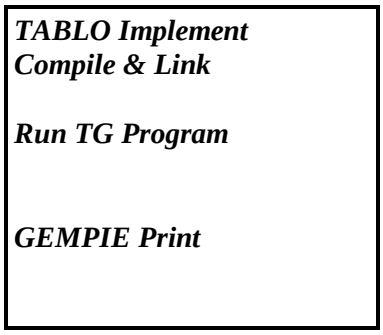
2.5.5 The Example Simulation using a TABLO-generated Program

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 or F77L3) 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 2.5.6) 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 - Look at the results

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

2.5.5.1 Step 1 - Implementing the model SJ using TABLO.

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.

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 3.3.2 below. For the present, we suggest that you take this on trust and continue working through Step 1.

[14] Another alternative is to use TABmate to implement your model by selecting instead
Simulation | TABmate Implement

2.5.5.2 Step 2 - Solve the equations of the model.

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 GEMPIE***.[15]

If there is an error, view the Log file.

There is no point in trying to look at the Solution file directly because it is a binary file, not a text file. GEMPIE will convert the solutions into a Print file that can be edited or printed out.

2.5.5.3 Step 3 - Look at the results using GEMPIE.

The *Go to GEMPIE* button takes you to the GEMPIE window.

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

The Solution file is already selected. Click on ***solutions?*** to see what solutions are available for printing. 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.]

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

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

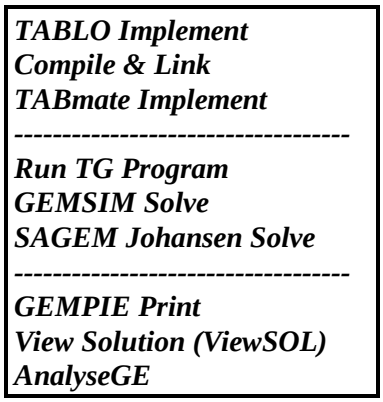
Now you can either jump to section 2.5.7, or you can read section 2.5.6 if you also wish to know about using GEMSIM for simulations.

---

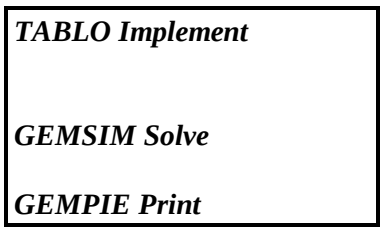
[15] The alternative ***Go to ViewSOL*** is equally useful for looking at the simulation results. This alternative is described in GPD-8.

2.5.6 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[16]



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



TABLO, GEMSIM and GEMPIE 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 - Look at the results

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

[16] 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.

2.5.6.1 Step 1 - Implementing the model SJ using TABLO.

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**.[17] This is all TABLO needs to implement the model.[18]

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 in 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.

2.5.6.2 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?

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

---

[17] This TABLO Input file is shown in full in section 3.3.2 below. For the present, we suggest that you take this on trust and continue working through Step 1.

[18] Another alternative is to use TABmate to implement your model by selecting instead
Simulation | TABmate Implement

[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 GEMPIE***.[19]

If there is an error, view the Log file.

There is no point in trying to look at the Solution file directly because it is a binary file, not a text file. GEMPIE will convert the solutions into a Print file that can be edited or printed out.

2.5.6.3 Step 3 - Print out the results using GEMPIE.

The *Go to GEMPIE* button takes you to the GEMPIE window.

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

The Solution file is already selected. Click on ***solutions?*** to see what solutions are available for printing. Click on the ***Ok*** button to continue.

Click on ***Run*** to run the program GEMPIE. If you carried out Example 1.5 above you will be warned that you are about to overwrite the file SJLB.PI5 and asked if you want to change the name of the output file; click on ***No***.

When GEMPIE finishes running, 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.]

Then select ***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.)

2.5.7 GEMSIM or TABLO-generated Program?

Readers with a Source-code version of GEMPACK 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.

We suggest that you now move ahead to section 2.7 "Interpreting the Results". [If you are interested in working 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.]

---

[19] The alternative ***Go to ViewSOL*** is equally useful for looking at the simulation results. This alternative is described in GPD-8

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 2.2.2 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 2.1 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 2.6.1 and the GEMSIM alternative in section 2.6.2.

2.6.1 The Example Simulation using a TABLO-generated Program

2.6.1.1 Step 1(a) - Run TABLO

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.)

```
<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**),[20]
- 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.

2.6.1.2 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 [21]

```
ltg sj
```

to do this. (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).

2.6.1.3 Step 2 - Run 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 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 ;
```

---

[20] 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.

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

(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 SJLB.CMF until section 2.8.1 below.

When the program is finished, you can check that the Solution file **sjlb.sl4** has been created.[22] This contains the numerical results of the simulation.

2.6.1.4 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 3)
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.]

[22] 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.

2.6.2 The Example Simulation using GEMSIM

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

2.6.2.1 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 2.6.1 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 2.6.1 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.

2.6.2.2 Step 2 - Run 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
```

2.6.2.3 Step 3 - Run GEMPIE

When GEMSIM has finished running, run GEMPIE exactly as in section 2.6.1.4 above to produce the GEMPIE Print file **sjlb.pi5**.

2.6.3 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.

2.7 Interpreting the Results

The results of the simulation, as given by GEMPIE, are shown in Table 2.7 on the next page. We think that you will find the table fairly easy to interpret. [Table 2.7 is actually a selected part of the Print file produced by GEMPIE (step 3 for the simulation).]

The results there imply that, if the supply of labor is increased by 10 per cent and the supply of capital is held fixed, then, for example,

1. the dollar value of total nominal household expenditure will increase by 5.8853 per cent from its pre-simulation value (this is the simulation result for 'p_Y', the percentage change in the levels variable 'Y'),
2. households will consume 6.8993 per cent more of commodity 2 than they did previously (the 'p_XH' result for commodity 2),
3. the price of commodity 2 will fall by 0.9486 per cent (the 'p_PC' result for commodity 2), and
4. the dollar value of household consumption of commodity 2 will rise by 5.8853 per cent (the p_DVHOUS("s2") result).

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 2.2.1a 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 beneath 5.8853 under p_Y 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).

Table 2.7: 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
```

Of course the updated values in (2), (3) and (4) above should be related since dollar value should equal price times quantity. Note that this is true since, from (2) and (3) above, the post-simulation price times the post-simulation quantity is

$$0.99051 \times 4.2760 = 4.2354$$

which is indeed the post-simulation dollar value in (4). This confirms that the solution shown in 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 endogenous and exogenous variables), 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.)

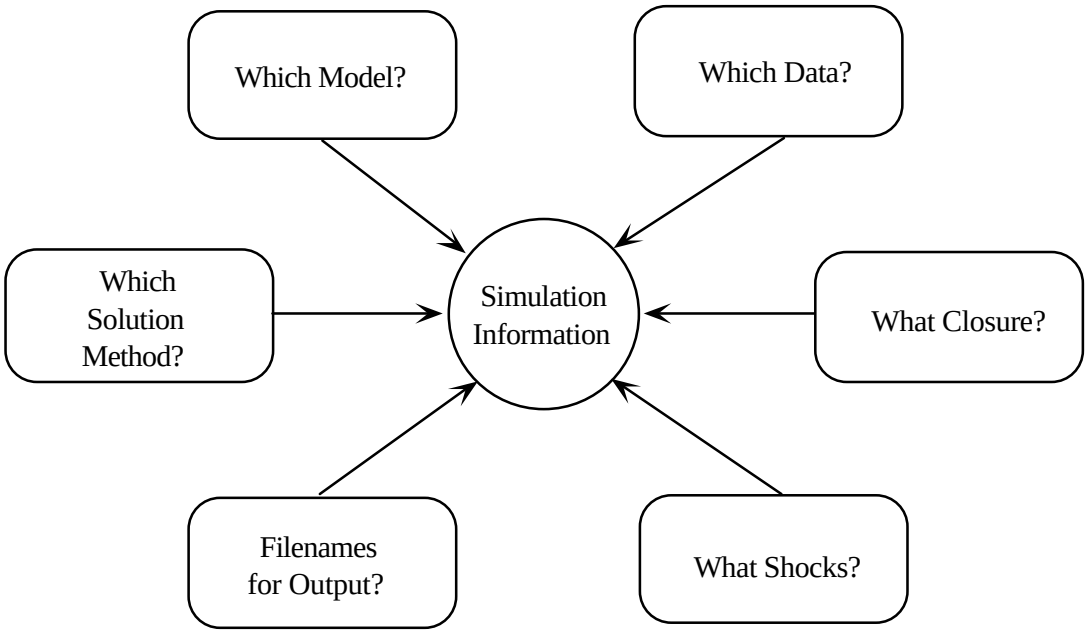


Figure 2.8 The Information Required to Specify a Simulation

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.

Of course, the instructions in this Command file must be prepared in advance.

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.][23]

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 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
```

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 2.2.1a above).

[23] 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 line

```
! Data files
```

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 comments 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 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 2.2.1a 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 uipdated 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.][24]

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

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. [25]

---

[24] 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.

[25] 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).

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 2.11.3 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 2.11.3 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 2.11.3 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.

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.

2.9 The Updated Data - Another Result of the Simulation

As you saw in the section 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. This data represents the state of the economy after the shocks, which is why the updated data is sometimes referred to as the **post-simulation data**.

GEMPACK provides easy ways of looking at data files. For example, Examples 2.1.1 and 2.1.2 of GPD-8[26] tell you how to use ViewHAR or SEEHAR to look at the base (or pre-simulation) data file SJ.DAT which is the starting point for the simulation above; of course, when you look at this file you see the numbers shown in Table 2.2.1a above.

You can use the same software and methods to look at the updated data in file SJLB.UPD. If you do this you will see the values in the Table 2.9 (which are shown to 3 decimal places).

		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 2.7 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.

[26] These examples are for WinGEM users. Command prompt users can look in section 3.1.1 of GPD-8. Note that ViewHAR is not available on Unix etc, but SEEHAR is.

2.10 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**[27]**,** 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 2.11.1 below.)

2.10.1 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 2.10.1: 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-

[27] 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.[28]

2.10.2 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 2.10.2.1. If you are working at the Command prompt, proceed as in section 2.10.2.2.

2.10.2.1 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 2.10.1 above)

```
use equations file sj ;
```

(ii) to create the Solution file called SJLBJ.SL4 (the Command file just says "sjlbj" but SAGEM will automatically add the standard Solution file suffix ".SL4")

```
solution file = sjlbj ;
```

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

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

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

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

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

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

---

[28] 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*** 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 2.10.3 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 2.10.3 below.]

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

Now skip ahead to section 2.10.3 where we discuss the results.

2.10.2.2 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 (its full name is usually SJLBJ.PI5), which you should print or look at in an editor.

Check that your results are the same as those shown in Table 2.10.3 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 2.10.3 below.]

This completes the Johansen simulation.

2.10.3 The Results of These Johansen Simulations

Some of the results from GEMPIE Print file SJLBJ.PI5 are shown in Table 2.10.3.

Page 1 of the results (these are on the second last page of the Print file) should be (approximately) as shown in Table 2.10.3. (Page 2 of the results gives the results for variables p_DVFACIN and p_DVHOUS, which we do not show in Table 2.10.3.)

The results shown in Table 2.10.3 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[29] more quickly (that is, using less computing resources) than one single, more accurate, multi-step simulation result.

[29] Indeed, the run of SAGEM in question can be said to produce the results of infinitely many Johansen simulations since it is easy to read off the (Johansen) result corresponding to any pair of increases/decreases in the supply of labor and capital. (Note that multi-step simulation results cannot be combined in this way.)

Table 2.10.3 : 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.
```

2.10.4 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 2.10.4 below. Notice that this uses the Equations file SJ.EQ4 and the Environment file SJXFAC.EN4 produced during the run of SJ.EXE or GEMSIM in section 2.10.1 above.

Figure 2.10.4: 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[30] 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 Johansen simulation relates to the updated data. If you are only carrying out a single Johansen simulation (perhaps just giving one set of shocks and are only concerned with the cumulative result), SAGEM will not produce the updated data whereas SJ.EXE or GEMSIM will.

[30] 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.

2.10.5 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 is

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

2.11 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.

2.11.1 The Linearized Equations of a Model

Johansen solutions are calculated by solving the linearized equations[31] 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 2.10.1), [32]

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. It is often useful to think of this matrix as a rectangular array or tableau[33] 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 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.

---

[31] 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.

[32] 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 2.1.1 or 2.1.2 above). This matrix C is often denoted by A(V) in DPPW.

[33] This explains the origin of the name TABLO.

Table 2.11.1: 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.

2.11.2 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 $\mathbf{C\,z = 0}$ in (1) above, becomes .

$$\mathbf{A\ z_1 = -D\ z_2} \qquad (2)$$

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 columns of the matrices A and D are just the columns of C corresponding to the endogenous and exogenous variables respectively. The shocks are the values to use for z_2. Once these are known, we have a system

$$\mathbf{A\ z_1 = b} \qquad (3)$$

to solve (where b is an n x 1 vector). It is the solution $\mathbf{z_1}$ of this matrix equation (3) which is the **Johansen solution** of the simulation.[34]

[34] 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.

2.11.3 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 2.11.3 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.

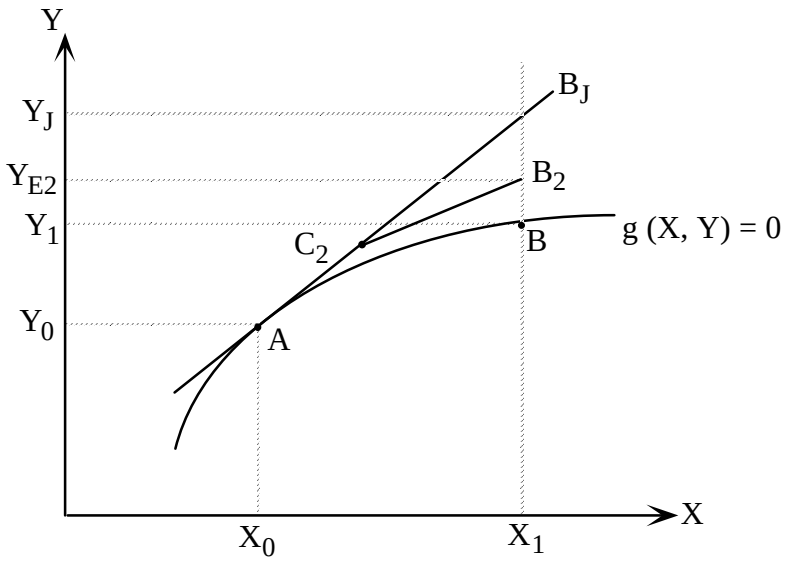


Figure 2.11.3 Illustration of Euler's Method

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 2.11.3), 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. 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 then to 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 2.2 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 [35]		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 that 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.
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), 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.)

For example, for the 1,2,4-step Euler results for household expenditure 'p_Y' reported above, the relevant line in the Extrapolation Accuracy file would be

p_Y 1 6.00000 5.94286 5.91412 5.88527 CX 4

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.)

[35] A 1-step Gragg calculation doesn't make much sense, so we have not shown a result for it.

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.

   (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 have been judged accurate to different numbers of figures.

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).

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 section 3.4. 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 3.3.2 is a mixed one (in the sense that it contains a mixture of linearized and levels equations). In sections 3.5.1 and 3.6.1 we describe alternative TABLO Input files for Stylized Johansen consisting only of linearized or levels equations respectively. In section 3.7 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 3.8.

In section 3.9 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 3.10 where we give you advice about building your own model by writing a TABLO Input file from scratch or by modifying an existing one.

Levels Form*	Linearized Form	
consumer demands $X_{i0} = \alpha_{i0} Y / P_i$	$x_{i0} = y - p_i$	i = 1,2
intermediate demands $X_{ij} = \alpha_{ij} X_j \prod_{t=1}^{4} P_t^{\alpha_{tj}} \left[\prod_{t=1}^{4} (\alpha_{tj})^{-\alpha_{tj}} \right] / [A_j P_i]$	$x_{ij} = x_{ij} = x_j - (p_i - \sum_{t=1}^{4} \alpha_{tj} p_t)$	i=1, ..., 4 j=1,2
price formation $P_j = \left[\prod_{t=1}^{4} (\alpha_{tj})^{-\alpha_{tj}} \right] \prod_{t=1}^{4} P_t^{\alpha_{tj}} / A_j$	$p_j = \sum_{t=1}^{4} \alpha_{tj} p_t$	j = 1,2
commodity market clearing $\sum_{j=0}^{2} X_{ij} = X_i$	$x_i = \sum_{j=0}^{2} [\frac{X_{ij}}{X_i}] x_{ij}$	i = 1,2
aggregate primary factor usage $\sum_{j=1}^{2} X_{ij} = X_i$	$x_i = \sum_{j=1}^{2} [\frac{X_{ij}}{X_i}] x_{ij}$	i = 3,4
numeraire $P_1 = 1$	$p_1 = 0$	
intermediate demands – dollar values $D_{ij} = P_i X_{ij}$	$d_{ij} = p_i + x_{ij}$	i=1, ..., 4 j=1,2
consumer demands – dollar values $D_{i0} = P_i X_{i0}$	$d_{i0} = p_i + x_{i0}$	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 α 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

$$D = PQ$$

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 3.8 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.[36] 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 3.5 and 3.6 respectively, we describe implementations based on exclusively linearized equations (section 3.5) and exclusively levels equations (section 3.6). 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.

---

[36] 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 3.3.2 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 2.2.1a in section 2.2.1 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 them must come

- the VARIABLEs (levels or linearized) occurring;
- the SETs used to describe the different arguments of variables;
- the data to be read;
- means of calculating pre-simulation values of any levels variables not read in as data (calculations are done via FORMULAs);
- means of calculating (via FORMULAs) 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 3.4 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 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 3.3.2 and then discuss the rest of this file in section 3.3.3 below.)

Consider first the very simple equation (E9) relating prices, quantities and dollar values of household consumption.

In the TABLO Input file this equation could be written as[37]

```
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,

[37] 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.

- 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 input.

For this equation to be meaningful, we must explain in the TABLO Input file all the names used in the equation.

The levels variables can be 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 names of the associated linear variables must also be used in preparing GEMPACK Command files (see sections 2.8.1 and 2.10.4 above); this is because the levels equations are linearized automatically by TABLO and so only the linear variables appear by the time you come to carry out simulations with your model.

The fact that SECT is a set with two sectors "s1" and "s2" in it can be indicated via the statement

```
SET SECT  # Sectors #   (s1-s2) ;
```

We must also indicate how pre-simulation values of the levels variables can be inferred from the data base. We can 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[38]

```
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[39] of the file containing the actual data.

---

[38] This explains why we have written the equation as shown rather than the more natural DVHOUS(i)=PC(i)*XH(i).

Secondly, consider the equation (E4) "price formation for commodities". This can be 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. That the values of these can be calculated from the data base can be communicated via the statements

```
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.

This introduces the main types of statements in a TABLO Input file, namely EQUATIONs, FORMULAs, READs, VARIABLEs, COEFFICIENTs, SETs and FILEs.

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. These statements can be added anywhere after the FORMULA giving the values of ALPHAFAC.

Each entity (VARIABLE, COEFFICIENT, etc) must be declared on the TABLO Input file before it is used in EQUATIONs and FORMULAs. This partly determines the order of the statements on the Input file. We suggest that you now look at the complete TABLO Input file for this model, as set out in section 3.3.2 below. 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 them 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 3.3.3.

---

[39] 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.

3.3.2 The TABLO Input File for the Stylized Johansen Model[40]

```
!------------------------------------------------------------------!
!                  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 ;
!------------------------------------------------------------------!
!      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 ! ;
```

[40] 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.

```
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 ! (E3.2.2) with i=1,2 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    ! (E3.2.2) with i=3,4 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-----------------------------!
```

3.3.3 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). We have introduced DEFAULT statements to 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 3.3.2 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. We will see non-parameter COEFFICIENTs and non-initial FORMULAs in section 3.5.1 below, when we 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

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 ;")

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 3.1.1, 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 3.4.3 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.

3.3.4 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; 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.

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 added to the levels name). 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_".)

3.3.5 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 3.3.2 above) containing a mixture of levels and linearized equations. We will introduce more information about the syntax and semantics in sections 3.5 and 3.6 below (where we describe alternative TABLO Input files for Stylized Johansen, firstly one containing just linearized equations and secondly one containing just levels equations).

3.4 Constructing Data Files

In preparing the TABLO Input file for a model, you work out how much data is required 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 which can be read by the GEMPACK programs containing these numbers. These files can be GEMPACK Header Array files (which are binary files) or GEMPACK text data files.

You can create GEMPACK text data files using your favourite editor or in a spreadsheet (see section 3.4.4). The syntax required for text files is introduced in the next subsection, section 3.4.1, 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, and there is a danger that data on them will be assigned to the wrong COEFFICIENTs if the order of your READ statements 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. The GEMPACK program MODHAR makes it fairly easy for you to create these files and to subsequently modify the data on them, while the programs SEEHAR and ViewHAR enable you to examine the data on these files.[41]

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 spread-sheet 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 subsection below.

3.4.1 Constructing the Data File for Stylized Johansen

As we 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 2.2.1a of section 2.2.1 above, namely

```
            ( 4.0   2.0 )                    ( 1.0   3.0 )
DVCOMIN     (             )          DVFACIN  (           )
            ( 2.0   6.0 )                    ( 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

[41] MODHAR is introduced in sections 3.4.1 and 3.4.2 below and is fully documented in chapter 3 of GPD-4. See section 2.2 of GPD-4 and also GPD-8 for information about ViewHAR.

header "<header>" longname "<long name>"

The header must agree with that specified for the relevant READ statement on the TABLO Input file (see section 3.3.2 above). 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 actual 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.[42]

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 (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 3.4.1 of
!  GEMPACK document GPD-1.
! The data are as set out in Table 2.2.1a 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
!
```

[42] 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.

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.[43]

These 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.

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. The file **SJDAT.TXT** shown above is usually supplied with GEMPACK. [If not, you will have to create this text file using a text editor.]

[Note that for small models such as Stylized Johansen, it is not really essential to have prepared the file SJDAT.TXT. This data could be entered from the terminal when running MODHAR. But we have used a file to illustrate the method applicable for larger models.]

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[44]
Standard input-output data for the Stylized Johansen model. ! History
**end             ! end the history
y                 ! Yes, this history is what I wanted
```

You may wish to check that the file SJ2.DAT has been created.[45] 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 Stored-input file to control the run since the responses required are easily predicted. (If you do prepare such a file and use the STI option when running MODHAR, you will not need the last line shown above, as explained in section 4.3.2 below.)

[43] 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.

[44] On some machines (not PCs) you will be asked to enter the date after this prompt about your name.

[45] 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.

Step 3 - Run a Simulation with Zero Shock to Add Set and Element Labelling

Ideally Header Array files 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 is described in chapter 2 of GPD-8), 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.[46]

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.[47] 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 Step 2 from part of section 2.5 or 2.6), 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.[48]

[46] Other ways of adding set and element labelling are described in section 5.3 of GPD-4.

[47] The file SJLABEL.CMF is usually supplied with GEMPACK.

[48] If you wish, you can follow the method in Example 2.1.1 or 2.1.2 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 Command prompt can use SEEHAR to see this difference.]

3.4.2 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 3.4.1 above) and then re-running MODHAR, or by running MODHAR to modify the data on the Header Array file directly.

Below we give a simple example of the second method, showing how you can use MODHAR to modify data on the Header Array file produced in section 3.4.1 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 2.2.1 above).[49]

If you run MODHAR and give the following responses, the new file (SJMOD.DAT) created will have this change incorporated in it.

```
<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
                (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
```

Example of Responses to MODHAR to Modify Data for Stylized Johansen

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 (see section 2.2 of GPD-4) can be used to modify data on a Header Array file. See the ViewHAR Help file for details.

[49] Note that the data base will still be balanced after this single change.

3.4.3 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 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. 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. Usually TABLO Input files for data manipulation have no EQUATIONs (and hence no VARIABLEs or UPDATEs) in them.

Below we give a simple example of a 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.

Example of a TABLO Input File to Carry Out Data Manipulation

```
!    Using TABLO to manipulate data !
! The next statement indicates that there are no equations here !
EQUATION (NONE) ;

SET COM # Commodities # (com1 - com10) ;
SET IND # Industries #  (ind1 - ind8) ;
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";
```

In this example, reading is done from a text file and writing is to a Header Array file. GEMSIM and TABLO-generated programs can also read from Header Array files and write to GEMPACK text data files.

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 (see section 2.2 of GPD-4) to aggregate data in different ways.

See section 4.15 of GPD-2 for more about the statement "EQUATION(NONE);" which is usually put at the start of TABLO Input files doing only data manipulation.[50]

3.4.4 Header Array and Text Data Files

In this subsection, we give a brief account of these files and the principal ways they are used in GEMPACK.

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-letter **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.

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

SUMHAR	For summarising the contents of a Header Array file
SEEHAR	For looking at the actual data on 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 SUMHAR and SEEHAR are given in chapter 4 of GPD-4, MODHAR is documented in chapter 3 of GPD-4, and ViewHAR (which is only available on Windows PCs) is introduced in section 2.2 of GPD-4 and in chapter 2 of GPD-8.

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** and then, using MODHAR much as in section 3.4.1 above, create a GEMPACK Header Array file

[50] EQUATION(NONE) is no longer needed with Fortran 90 TABLO.

containing the data. If you subsequently wish to modify the data, you can use SEEHAR to write it out in CSV format, import it into your spreadsheet and modify the data there, then write out the modified data in CSV form and rerun MODHAR to create a Header Array file containing the modified data. Note that, even if you are running GEMPACK on a mainframe computer and your spreadsheet program is on a PC, you can still do this since transferring text files between PCs and mainframes is easy (using FTP, for example).

One way of preparing reports on simulations carried out with your model is to first convert the simulation results (stored on a GEMPACK Solution file) to a text file (via the GEMPACK program SLTOHT described in chapters 8 and 9 of GPD-4) and then import this text file into a spreadsheet or graphing package in order to prepare tables or graphs of your results.

Text data files can also 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[51] 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 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.

Note also that GEMSIM and TABLO-generated programs can be used to write data files (see, for example, section 3.4.3 above). These programs can write data text files. To do this, the FILE statement declaring the logical file to be written 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.

[51] 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.

A summary of which programs to use in converting files from one type to another is given below in Table 3.4.4. It includes references to sections of this and other documents where these programs are described in more detail.

Table 3.4.4: 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'	3.4.1 Also GPD-4
or				
Text file	Header Array file	ViewHAR		ViewHAR help
Header Array file	Text file(CSV) (row order) (column order)	SEEHAR	SS ROW COL	GPD-4 chapter 4
Solution file	Header array file	SLTOHT	-	GPD-4 chapter 8
Solution file	Text file(CSV) (row order) (column order)	SLTOHT	SS,SSS SIR SIC	GPD-4 chapter8,9
Solution file	Text file(CSV)	ViewSOL		ViewSOL help
Text file (CSV)	Spreadsheet data	Spreadsheet	-	-

3.5 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**.[52]

We illustrate this by giving in full in section 3.5.1 below such a TABLO Input file for Stylized Johansen.

In comparison with the mixed TABLO Input file for Stylized Johansen in section 3.3.2 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 2.11.3 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 sections 3.5.3 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 3.5.1 and then discuss noteworthy features of it in section 3.5.2 below.

Advice about linearizing equations by hand can be found in section 9.2 in GPD-2.

[52] Indeed, in releases of GEMPACK prior to Release 5.0 (April 1993), these were the only kind of TABLO Input files allowed.

3.5.1 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
  # Intermediate demands for commodity i by industry j #
    (all,i,SECT) (all,j,SECT)
     p_XC(i,j) = p_XCOM(j)  -
   [ p_PC(i)  - SUM(t,SECT,ALPHACOM(t,j)*p_PC(t))
                      - SUM(f,FAC,ALPHAFAC(f,j)*p_PF(f)) ]
    ! This is (E3.2.2) (i=1,2; j=1,2) in DPPW !  ;

EQUATION  Factor_inputs  # Factor input demand functions #
   (all,f,FAC) (all,j,SECT)     p_XF(f,j) = p_XCOM(j)  -
 [ p_PF(f)  - SUM(t,SECT,ALPHACOM(t,j)*p_PC(t))
                     - SUM(g,FAC,ALPHAFAC(g,j)*p_PF(g)) ]
    ! This is (E3.2.2) (i=3,4; j=1,2) in DPPW !  ;

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---------------------------------!
```

3.5.2 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 3.5.1. This is because of the convention that all TABLO Input files are assumed to begin with defaults appropriate for linearized TABLO Input files,[53] 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 3.3.2 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 #
```

[53] 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.

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 Table 2.7 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)
```

[These ORIG_LEVEL qualifiers are not necessary in the mixed TABLO Input file SJ.TAB shown in section 3.3.2 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 3.3.2 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.

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 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 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. 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.[54]

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,

[54] There are other kinds of UPDATE statements called CHANGE UPDATEs. They are less commonly needed and are documented in section 3.5.3 below.

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,

   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.

3.5.3 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 3.5.3 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.[55] All three UPDATE statements in the linearized TABLO Input file for Stylized Johansen are of this form (see section 3.5.1 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 3.5.1). Note

[55] 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.

also that only COEFFICIENTs whose values are READ or assigned via a FORMULA(INITIAL) in the TABLO Input file must 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. We illustrate this in the example below. Advice about linearizing equations by hand can be found in section 9.2 of GPD-2.

Example of an UPDATE (CHANGE) statement

Consider a commodity with base level price P, a quantity Q of which is sold and which is taxed at an *ad valorem* rate of T per cent. Suppose that read from the data base are COEFFICIENTs V representing the pre-tax dollar value and W representing the post-tax dollar value of sales of this commodity. Suppose also that VARIABLEs in the TABLO Input file include p_P and p_Q being the percentage changes in P and Q respectively and also c_T being the CHANGE (not percentage change) in T.[56]

Clearly the relevant levels equations are

```
V = P*Q                                    (1)
W = P*Q*(1+T) = V*(1+T)                    (2)
```

Updating V is easy since clearly case 2 above applies. Here we work out an appropriate UPDATE statement for W, which must follow case 3 above (since cases 1 and 2 do not apply).

To do this amounts to linearizing equation (2) which we do by partial differentiation in the usual way (see section 9.2 of GPD-2). Using **d** in front of quantities to denote differentials or changes, we obtain

```
dW = dV*(1+T) + V*d(1+T)
   = dV*(1+T) + V*dT              (since d(1+T) = dT)
   = dV*(1+T) + V*c_T                (3)
```

since c_T is the VARIABLE representing the change in T. But, linearizing (1) similarly by partial differentiation and using the fact that changes dE and percentage changes pE in a quantity E are related via dE = E*pE/100, we obtain

```
dV = P*dQ + Q*dP = P*Q*p_Q/100 + Q*P*p_P/100
                 = P*Q*[p_P + p_Q]/100
                 =  V*[p_P + p_Q]/100.
```

If we substitute this into (3) we obtain

```
dW = W*[p_P+p_Q]/100 + V*c_T
```

as the expression for the change in W. Thus the appropriate UPDATE statement for W is

UPDATE (CHANGE) W = W*[p_P+p_Q]/100 + V*c_T ;

Of course if the VARIABLE in the TABLO Input file were p_T, the percentage change in T, rather than c_T the change in T, then we would have c_T=T*p_T/100, in which case an appropriate UPDATE statement for W would be

UPDATE (CHANGE) W = W*[p_P+p_Q]/100 + V*T*p_T/100 ;

More details about Updates can be found in section 4.11 of GPD-2.

---

[56] Since T is usually positive but may also be negative if the tax is in fact a subsidy, it may be unwise to include the percentage change in T as a VARIABLE. In this case the VARIABLE c_T would be declared in the TABLO Input file via a VARIABLE(CHANGE) statement, as explained in section 3.3.4 above.

3.6 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 3.6.1. 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.

3.6.1 Levels TABLO Input File for Stylized Johansen

The main difference from the mixed TABLO Input file shown in section 3.3.2 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). 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.[57]

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 for the quantity called W(j) there. This has been introduced to simplify the "intermediate demands" and "price formation" equations. W(j) is most naturally a product of several quantities. However, although TABLO recognises SUMs, it does not allow products, which would be naturally expressed via

PROD(<index>, <set>, <expression>).

Accordingly, we have converted the PROD implicit in the expression for W(j) into a SUM by taking logarithms of both sides.

The full levels TABLO Input file is shown below.

---

[57] 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.

```
!-----------------------------------------------------------------!
!                  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 accurary.
!

!   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 Extra (all,j,SECT)

 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-------------------------!
```

3.7 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* (1993a). 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.

3.8 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.[58]

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) proceed

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.
- when you carry out simulations by running GEMSIM or the TABLO-generated program from your model, you must refer to variables on the associated linearized file rather than the levels variables. (For example, when working with the mixed version of Stylized Johansen in section 3.3.2 above, you must say the p_XFAC is exogenous, not XFAC.)

Other features of this conversion are explained in section 2.2 of GPD-2.

[58] Actually this is not entirely accurate. If the levels VARIABLE is declared via a VARIABLE(LEVELS,CHANGE) statement (see section 3.3.4), the associated linear VARIABLE has "c_" at the start.

3.9 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 2.11.1 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 3.8 above). All condensation is done in relation to these linearized equations and so involves only linear variables.

3.9.1 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)].
```

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 4.3.2 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.

Example - Condensing Stylized Johansen

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 3.3.2 above, they suggest substituting out the variables p_XH, p_XC and p_XF using the 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. [The GEMPACK mixed implementation of the uncondensed model is of size 27 x 29, as shown in Table 2.11.1 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.]

We suggest that you make these substitutions with Stylized Johansen and then carry out a simulation with the resulting condensed model. Perhaps carry out the same simulation as in section 2.2 above and compare your results with those there. Of course, the results for endogenous variables remaining in the condensed system should be the same, except for possible rounding errors. Your responses in running TABLO could be as follows.

```
                  (First the Check)
<carriage-return> ! Default options
sj                ! Name of TABLO Input file
sjcond            ! Name of Information file ('COND' for condense)
                 (Next Condensation)
c                   ! do condensation
s                   ! substitute
p_xh                ! variable p_XH (case does not matter here)
house               ! using equation House
s                   ! substitute
p_xc                ! variable p_XC
comin               ! using equation Comin
s                   ! substitute
p_XF                ! variable p_XF
facin               ! using equation Facin
e                   ! exit from condensation
                 (Finally, Code generation)
a                   ! Automatic Code generation
pgs                 ! Prepare output for GEMSIM
<carriage-return> ! other default code options
sjcond            ! Name of GEMSIM Auxiliary files
```

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 2.5 and 2.6 above).[59]

Then continue with Steps 2 and 3 as in section 2.5 and 2.6 above. However, you may find it best to use SJCOND (rather than SJ) for the names of various files, to avoid confusion with the corresponding files for the uncondensed model. Of course you must not specify p_XH to be printed when running GEMPIE (see section 2.6.1.4 above) since p_XH is not present in the condensed version of the model.

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.

3.9.2 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

[59] Indeed, 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 2.5 and 2.6. Similar considerations apply to our choice of SJCOND as the Information file name.

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 3.9.1 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) to have the values of the variable in question calculated (by backsolving). Backsolving, which is done at each step of a multi-step simulation, is carried out after the LU decomposition and 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.

Example - Stylized Johansen
We suggest you carry out again the condensation of Stylized Johansen described in section 3.9.1 above, but this time indicate that you may want to backsolve for variables p_XH and p_XC. To do this, proceed as above but replace the relevant two 's' responses by 'b'. When you run GEMSIM, if you select all available variables to be cumulatively-retained endogenous, you will see that the values of the backsolved variables appear on the Solution file (and on the Extrapolation Accuracy file).

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) 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.

3.9.3 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, 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.[60]

3.9.4 Stopping and Restarting TABLO

Although it is usual to carry out all processing (Check, Condense, if required, and Code generation) of a TABLO Input file in a single run of TABLO, it is possible (and sometimes desirable) to stop after one of these stages and, later, to rerun TABLO starting from where you left off. When you stop after one of these stages, TABLO saves the results so that, when you resume, you do not have to repeat the processing already carried out. The savings are only worthwhile for very large models; for small and medium-sized models it is best to always start from the Check. [Indeed, computers have become so much faster in the years since TABLO was first introduced that we now see little value in re-starting TABLO. We recommend that you always start from the Check stage.]

For example, for a very large model, it may be desirable to carry out different condensations for different groups of simulations. In this case, it is a good idea to exit from TABLO after the Check. Then restart TABLO (telling it to start at the Condensation stage), carry out the first condensation and then generate code for this version of the model. Finally, repeat this for the second condensation.

When you exit from TABLO after the Check or Condensation stages, TABLO saves the results on two binary files (a TABLO Record file and a TABLO Table file). Together these make up what TABLO refers to as an **implementation**.

These two files take their names from that of the Information file but they have different suffixes; the TABLO Record file usually has suffix '.TBR' while the TABLO Table file usually has suffix '.TBT'.

You can further condense a model that has been partly condensed if you exited from TABLO after Condensation the first time. Simply start again at Condensation and carry out the extra condensation actions (using the implementation saved after the first condensation).

---

[60] 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.

Example

Suppose you have a large model with TABLO Input file MODEL.TAB and you want to have two different condensations. First run TABLO, giving the inputs shown below, exiting after the Check.[61]

User Input to Carry Out Just the Check

```
<carriage-return>          ! Default options
model                      ! TABLO input file name
model                      ! Information file and Implementation name
e                          ! Exit after the Check
```

After this you will have files MODEL.TBR and MODEL.TBT.

Then you can carry out the first condensation as follows. Notice that you tell TABLO to start at Condensation by selecting option 'F2' from the options presented at the start of TABLO.

User Input to Do Just Condensation and Then Code Generation

```
F2                    ! Start at Condensation
<carriage-return>     ! End of option selection
model                 ! Implementation name (see above)
                    (TABLO checks to see that MODEL.TBR and MODEL.TBT
                    exist and have the expected form.)
model_c1              ! Information file for condensation 1 (hence 'c1')
c                     ! Start Condensation
s                     ! Substitute (for example)
                    (We omit the actual condensation actions, which
                    may consist of several substitutions and/or
                    omissions. When the last one has been done,
                    continue as shown below.)
e                     ! Exit from Condensation
a                     ! Go on to Code generation
pgs                   ! Prepare output for GEMSIM
<carriage-return>     ! Other default code options
model_c1              ! Name of GEMSIM Auxiliary files
```

Note that the name of the GEMSIM Auxiliary files (or of the TABLO-generated program if you choose that option) should reflect the condensation you have carried out, which is why we used the name 'MODEL_C1' above.

Then, for the second condensation, proceed as above except that you should choose different names (perhaps 'MODEL_C2') for the Information file and GEMSIM Auxiliary files or TABLO-generated program produced.

Note that you can only start at Condense or Code generation if you have already carried out and exited after the previous stage(s) and have retained the TABLO Record and Table files produced. Otherwise you will have to start again from the Check.

[61] Alternatively you could tell TABLO to stop after Check by selecting option 'L1' (last stage is Check) as your first response. Then you would not need the 'e' response at the end.

3.10 Creating the TABLO Input File 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 1.2 of GPD-2.

Alternatively, you can create a TABLO Input file for your model from scratch. Suggestions about this can be found in section 1.3 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. A simple example of this can be found in section 2.1.13 of GPD-8.

CHAPTER 4

4. Common Features of GEMPACK Programs

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 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 4.1), they all allow comments starting with a single exclamation mark in input from the terminal (see section 4.2), and they all offer standard options (such as the ability to direct output to a Log file) as explained in sections 4.3 and 4.4. It is possible to specify various files (including Command files, Stored-input and LOG files) on the command line: details are given in section 4.5. The procedure by the programs manage memory available is described in section 4.6 while section 4.7 describes the GEMPACK error reporting scheme. Section 4.8 is about file names, file suffixes and GEMPACK file types. Section 4.9 tells you what files names are allowed on different systems.

Sections 4.1 to 4.5 will be mainly of interest to those who run GEMPACK programs from the Command prompt (either interactively or via Stored-input files), though 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.

4.1 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".

On most systems (but not on Unix systems), the case in which you enter file names makes no difference. Some input to GEMPACK programs (such as the "verbal description" of a simulation) is case-sensitive, of course. (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.)

4.1.1 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.

4.2 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 4.3 below). We 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, 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.

4.3 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**.

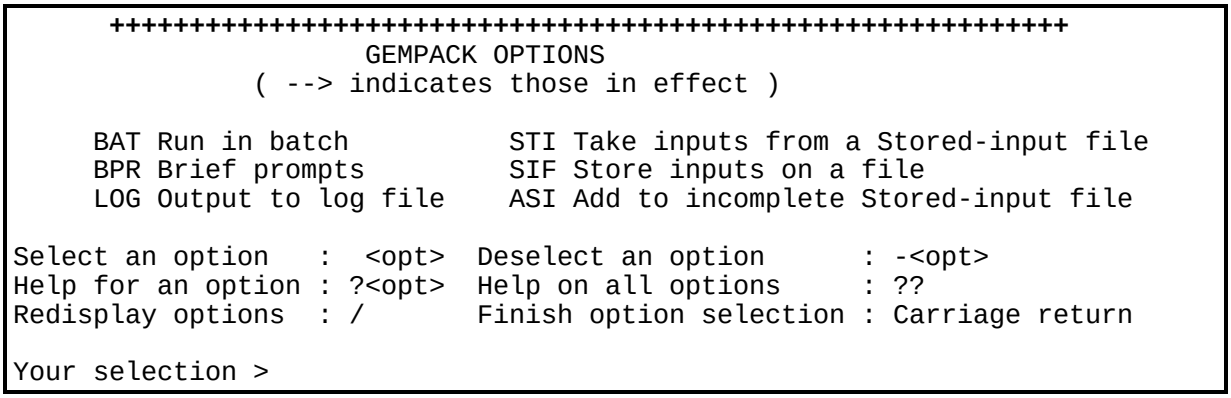
Most operating systems provide their own ways of running programs in batch mode. (For example, on DOS PCs or on Unix machines, you can use input redirection such as

```
sagem < sjlb.sti
```

to run the program SAGEM, taking input from file 'sjlb.sti'.[62])

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:

[62] Provided your SAGEM program is made with Release 6.0 or later of GEMPACK, we suggest that you use **–sti** as in "sagem -sti sjlb.sti" (see section 4.5 below) rather than "<".



Options Screen for GEMPACK Programs

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.

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 4.3.3.

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 4.5.

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 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. 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 modification. 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 the option 'sti' has 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.)

4.3.1 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.

4.3.2 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 selected 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.[63]

4.3.3 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** [64], 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 the command

```
        log only = <filename> ;
```

   or

```
        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

---

[63] 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.

[64] Option **cmf** is the option which indicates that you wish to take input from a GEMPACK Command file (see sections 2.2.3, 2.3.1 and 2.4 above). It is only available when running SAGEM, GEMSIM or a TABLO-generated program.

```
log file|only = yes ;
```

4.3.4 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[65]). 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.

4.4 Other Program Options

The final 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.

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.

---

[65] This early output was not recorded in the LOG file in Release 5.1

4.5 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 and then indicating "cmf" and then "sjlb.cmf". This option "-cmf" can be used with GEMSIM, TABLO-generated programs and SAGEM.

You can also use **`-sti  <STI-file-name>`** to specify a Stored-input file for any program. For example, the command

modhar -sti modhar1.sti

will run MODHAR taking inputs from the Stored-input file MODHAR1.STI.

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 in some circumstances.

The command line options described above are available on Windows PCs (under DOS) and on Unix machines.[66]

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"
```

4.5.1 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". 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. [Actually there may be

[66] 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.

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.

4.5.2 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.[67] 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.

4.6 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. Fortran has two well-defined standards, Fortran 77 and Fortran 90, and can be used on various computers with only minor changes to the Fortran code. 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.

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

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.

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

In particular, Windows PC users with Lahey F77L3 are encouraged to upgrade to Lahey LF90 or LF95 as soon as possible. F77L3 does not perform well under the later Windows operating systems, notably Windows NT and 2000. In order to reduce our code maintenance work, we expect **not to support Fortran 77** (including F77L3) in the next release of GEMPACK after Release 7.0.

Please see chapter 13 of GPD-3 for information on memory management, and instructions on how to increase parameters if you are using Fortran 77.

[67] This was introduced in Release 7.0.

4.7 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.[68]

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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.

[68] The actual message, which is longer that the one shown here, includes email addresses to use to send the files to us.

4.8 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.

4.8.1 Files with System-determined Suffixes

The file names of these files are usually chosen by you. 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 4.8.1: File Type with System-Determined Suffixes

File type	Suffix[69]
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

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
```

[69] On Unix machines they are all lower case.

4.8.2 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.

Table 4.8.2: Commonly Used Suffixes for Other Files

File type	Usual Suffix(es)[70]
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
Extrapolation Accuracy file	.XAC
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.

4.8.3 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 section 3.4), where each array of data is identified by specifying its 4-character header. Text data files used in GEMPACK must follow a standard format which was introduced in section 3.4 and is documented fully in chapter 6 of GPD-4.

[70] On Unix machines they are usually all lower case.

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

4.8.4 Why So Many Files?

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 2.10.1 above).
- The **Extrapolation Accuracy file** can be used to tell if the solution produced is sufficiently accurate for your purpose (see section 2.11.3). (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.

4.9 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 be an allowed one. For example, if you are working with F77L3 MODHAR, you may ask it to create a file called SJ2.DAT. However if you are running inside the directory "C:\temp files\sj" you would not be able to create the file since the "temp files" part of the directory name is not allowed with F77L3 GEMPACK programs (see section 4.9.3 below).

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.]

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

4.9.1 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.

4.9.2 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).[71] The allowed characters were limited also to those currently allowed

[71] 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 deafult.] However spaces are not allowed in file or direcrory names and the allowed characters are restricted to those allowed currently with F77L3 programs.

under F77L3 – see section 4.9.3 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, NT and 2000.[72]

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 "%" so we advise you to not to use this character 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.

4.9.3 File Names on Windows PCs – Programs Made with F77L3

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.

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.

4.9.4 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,

---

[72] File and directory names on these systems cannot include any of the following characters:
\ / : * ? " < > |

solution file = “my sj” ;

See section 2.4.1 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 4.5), 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”

CHAPTER 5

5. A Guide to the Rest of the Documentation

There are currently seven GEMPACK documents which are referred to by their GPD numbers[73] :

GPD-1, GPD-2, GPD-3, GPD-4, (no GPD-5), GPD-6, GPD-7, GPD-8.

Their full titles are given in chapter 8.

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 has been rewritten and reordered:

- Material related to the program TABLO and 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 DOS/Windows PCs with Lahey Fortran,* and
- GPD-7 *Installing and Using the Executable-Image Version of GEMPACK on DOS/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.

---

[73] 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.

5.1 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 (1993b).

Features of GEMPACK which are useful in dealing with large models, especially multi-regional or intertemporal ones, are discussed in Harrison *et al* (1996).

CHAPTER 6

6. 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.

6.1 Release 6.0 (October 1998) and Release 6.0-001 (March 1999)

6.1.1 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.[74]

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

It is now possible to ask GEMSIM or a TABLO-generated program to calculate and report pre-simulation and post-simulation levels values as well as the usual percentage change results. To do this, you may need to add appropriate information in your TABLO Input file to tell TABLO what are the pre-simulation levels values of selected linear variables. For example, in the linearized TABLO Input file SJLN.TAB for the Stylized Johansen model (see section 3.5.1), add the qualifier "(ORIG_LEVEL=DVCOM)" when declaring the variable p_XCOM, as in

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

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

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

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

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

Then, when you run a simulation, GEMSIM or the TABLO-generated program will report the pre-simulation, post-simulation levels values of XCOM and PCOM, also the change in them, as well as the percentage change p_XCOM in XCOM and p_PCOM in PCOM, for each commodity. See section 4.5.5 of GPD-2 for details.

---

[74] 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 7.0) versions of the GEMPACK documents.

(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 4.5 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 1.2.6. See also section 1.2.

(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 5.5.5 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.

6.1.2 New Features in Relase 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.[75]

(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.
(ii) if the relevant "range test ..." is set at "YES", then the simulation ends with a fatal error after

[75] 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 7.0) versions of the GEMPACK documents.

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 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 "tnas = yes ;". When this is in operation, Assertions are not checked.
(ii) The statement "assertions = yes ;" is equivalent to the statement "tnas = 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 uaually 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.

6.1.3 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.

6.2 Release 5.2 (September 1996)

6.2.1 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.[76]

(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.]

---

[76] 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 7.0) 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 4.11.7 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 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.

6.2.2 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).

6.3 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.

7. 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.

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. 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 (1993b), '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.

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.

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.

8. GEMPACK DOCUMENTS[77]

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.

[77] 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".

9. INDEX

—<—

<cmf>
  Use in Command files, 2-45

—A—

Absorption, 6-132
Accuracy, 2-57
Address
  Contact, 1-15
AnalyseGE, 1-6, 1-10
  Incentive to backsolve, 3-105
ASCII file, 4-119
Auxiliary Statement,Table files, 2-36

—B—

Backsolving, 3-104, 3-105
  AnalyseGE, 3-105
Batch operation, 4-110, 4-112
Binary files
  Use in GEMPACK, 4-119
Bug
  Reporting, 4-117

—C—

C matrix, 2-55
  Condensation, 3-102
Calibration, 3-95
Case sensitive, 3-92, 4-109
  TABLO, 3-73
Case sensitive file names, 4-121
Centre of Policy Studies
  Address, 1-15
Change UPDATE, 3-93
Change variables, 3-75
Characters
  Chinese, 4-122
  Non-English, 4-122
Check
  TABLO, 3-102, 3-106
Chinese characters, 4-122
Closure, 2-36, 2-39
  invalid, 1-14
-cmf option, 4-115, 6-131
Code
  TABLO, 3-102, 3-106
COEFFICIENT
  TABLO, 3-74
Command file, 2-36
Command line options
  Available on Unix machines, 1-13
  available under DOS, 4-115
  available under Unix, 4-115
  -cmf, 4-115, 6-131
  -lic, 1-13, 4-116
  -log, 4-115, 6-131
  -lon, 4-115, 6-131
  -los, 1-13, 4-115
  only under DOS, 6-131
  -sti, 4-115, 6-131
Comments, 4-110
  Mapping files, 6-130
  Text files, 3-79
Compile, 2-36
Computation time
  Extrapolation, 2-58
  TABLO, 3-106
Condensation, 3-102
  Example, 3-103
Condense
  TABLO, 3-102, 3-106
Condensing twice, 3-106
Contact
  GEMPACK, 1-15
CSV format, 3-83, 3-84
Cumulative results, 2-53
Cumulatively-retained endogenous
  Backsolves, 3-105

—D—

Data
  Post-simulation, 2-47
  Pre-simulation, 2-47
  Updated, 2-47
Data bases, 1-11
  Converting Header Array files, 1-11
Data file, 3-77, 4-119
Data files
  Header Array, 3-77, 3-83
  Text, 3-83
Data manipulation
  TABLO, 3-82
Decomposing simulation results, 1-9
Default response, 4-109
Demonstration Version, 1-3, 1-4
  File names, 4-121
Different machines, 1-1
Directory
  Invalid, 2-26
  No spaces, 2-26
DISPLAY
  TABLO, 3-74
Documentation, 5-125
DPPW
  Reference, 2-18

—E—

Eliminated variables, 3-105
Empty Sets, 6-130
Endogenous, 2-17
Environment file, 4-120
EQUATION
  TABLO, 3-67, 3-74
Equation block, 2-55
EQUATION(NONE) statement, 3-83
Equations, 2-55

Linearized, 2-17
Equations file
Connection to Equations Matrix, 2-55
Contents, 2-55
Production and use, 4-120
Reason for the name, 2-55
Equations matrix, 2-55
Condensation, 3-102
Error trapping
Invalid arithmetic, 1-12
Logarithms, 1-12
Overflow, 1-12
Powers, 1-12
Euler's method, 2-57
Exclamation mark, 4-110
Executable image, 2-20
Executable-image licence limits, 1-4
Executable-image licences, 1-3
Limited, 1-3
Unlimited, 1-4
Executable-image version
Limited, 1-3
Unlimited, 1-4
Executable-image Version
File names, 4-121
Executable-image Versions, 1-3
Exogenous, 2-17
Extrapolation, 2-57
Extrapolation Accuracy file, 2-46, 4-120
Contents, 2-58
Extrapolation Accuracy Summary, 2-59

—F—

F77L3, 4-116
FAQs, 1-15
FILE
Logical name, 3-74
TABLO, 3-74
File names, 4-121
Allowed, 4-121
Case sensitive ?, 4-121
Demonstration Version, 4-121
Depend on compiler, 4-121
Depend on operating system, 4-121
Executable-image Version, 4-121
Invalid, 2-26
LF90 and LF95 programs, 4-121
Long, 2-26, 4-122
No trailing spaces, 4-121
FORMULA & EQUATION, 3-74
Fortran 77
Support may be phased out, 1-11
Fortran 90
Memory management, 4-116
Fortran compiler, 1-3
Fortran compilers
F77L3, 4-116
Fortran 77, 4-116
Fortran 90, 4-116
LF90, 4-116
LF95, 4-116
Unix, 4-116
Windows PCs, 4-116
Frequently Asked Questions, 1-15
FTP, 3-84
Functions
CUMNORMAL, 1-14
NORMAL, 1-14
RANDOM, 1-14

—G—

GEMPACK
Versions, 1-3
World Wide Web, 1-15
GEMPACK Documents, 5-125
GEMPACK licence, 4-116
GEMPACK licences, 1-3
Executable-image licence limits, 1-4
GEMPACK programs, 1-2
GEMPACK-L
Mailing List, 1-15
GEMPIE
Example, 2-37, 2-39
Print file, 2-20, 2-22, 2-37
GEMSIM
How to run, 2-39
GEMSIM Auxiliary files
Meaning, 2-22
GEMSIM Statement, Table files, 2-38
Gragg's method, 2-57

—H—

Harrison, Paul, 1-16
Header, 3-83
Header Array files, 3-83
Creating, 3-77
Modifying, 3-77
Headers
Reserved, 3-83
Help
Online, 4-114
Horridge, Mark, 1-16
How much data information, 3-77

—I—

Impact Project, 1-15
Implementation, 2-17, 3-106
Increasing Parameters, 4-116
Individual column results, 2-51
Information file, 2-36, 2-38
Interactive operation, 4-110, 4-112
Internal program error, 4-117
Introductory licence, 1-5
When required, 1-5
Invalid closure, 1-14
Invalid input
ASI, SIF, BAT, 4-112

—J—

Johansen simulation, 2-48, 2-56
Johansen solution
Meaning, 2-56

—K—

Keyword, 3-67, 3-73

—L—

Labelling information, 3-73
Large Models
  Condensation, 3-102
Large-simulations licence, 1-5
  When required, 1-5
Levels equations, 2-18, 3-63
Levels variables, 2-18, 3-75
LF90, 1-11, 4-116
LF95, 1-11, 4-116
LF95 Express, 1-11
Licence
  Introductory, 1-5
  Large-simulations, 1-5
Licence file, 4-116
Licences
  Executable-image licence limits, 1-4
Licences for GEMPACK, 1-3
Limited Executable-image version, 1-3
Limited Executable-image version licences, 1-3
Linearized equations, 3-63
Linerarized equations, 2-55
Log file, 2-46, 4-111
  Error, 4-114
  Terminal output, 4-113
-log option, 4-115, 6-131
Logarithms
  Error trapping, 1-12
-lon option, 4-115, 6-131
Long file names, 1-11, 2-26, 4-122
  Do not use "%", 4-122
  Only use "simple" characters, 4-122
Longname
  Header Array file, 3-83

—M—

Macro variable
  Meaning, 2-25
Mailing List
  GEMPACK-L, 1-15
Matrix
  Sparse, 2-56
Memory Allocation
  Fortran 90, 4-116
Midpoint method, 2-57
Model examples, 1-7
Model Representation
  Levels,Mixed,Linear, 3-100
Model users, 1-6
MODHAR, 3-81, 3-83
Multi-step solution, 2-55, 2-57

—N—

Non-linear equation, 2-57
Normal distribution, 1-14

—O—

Omitting variables, 3-106
Online Help, 4-114
Options
  GEMPACK BPR, 4-114
  GEMPACK Basic, 4-111
  GEMPACK BAT, 4-112
  GEMPACK LOG, 4-111, 4-113
  GEMPACK SIF, ASI, 4-111
  TABLO Code PGS, 2-22
  TABLO Code WFP, 2-20
Options menu
  GEMPACK programs, 4-111
Overflow, 1-12

—P—

Parameters
  In equations, 3-64
  Program, 4-116
Percentage change, 2-18
Percentage change variables, 3-75
Post-simulation data, 2-47
Powell, Alan, 1-16
Powers
  Error trapping, 1-12
Pre-simulation data, 2-47
Print file
  Meaning, 4-119
Program error, 4-117

—Q—

Quantifier, 3-68
Quantifier declaration, 3-73

—R—

Random numbers, 1-14
Range checking, 1-12
READ
  TABLO, 3-74
Reading data, 3-77, 4-119
Reducing model size
  Condensation, 3-102
Reporting program errors, 4-117
RunDynam, 1-10
RunGDYN, 1-10
RunGEM, 1-6
RunMONASH, 1-10

—S—

SAGEM, 2-48
  Backsolved variables, 3-105
Scalar variable
  Meaning, 2-25
SEEHAR, 1-11, 3-83, 3-84
Sensitivity analysis, 1-6. *See* Systematic sensitivity analysis
Set
  Empty, 6-130
SET

TABLO, 3-73
Set and element labelling, 3-80
Set mappings, 1-10
Shock
Target, 1-14
Shocks
From Header Array files, 1-11
Simulation
Meaning, 2-17
Steps, 2-19
Stylized Johansen example, 2-24
Simulation results
Analysing, 1-6
Simulations
Several at once, 2-48
SLC file, 1-10
SLTOHT, 3-84
Solution
Computation time, 2-58
Solution file, 2-20, 2-22, 2-37
Solving models, 2-17
Source-code versions, 1-3
Spaces in file names, 1-11
Spreadsheet, 3-83
Separator, 3-84
SSA. *See* Systematic sensitivity analysis
Statements in TABLO Input file, 3-73
-sti option, 4-115, 6-131
Stored-input file, 4-110
Meaning, 4-110
Terminal output, 4-113
Stylized Johansen model
Condensation, 3-103
Data, 3-66
Description, 2-24
Equations, 3-63
Equations matrix C, 2-55
GEMSIM simulation, 2-39
SAGEM simulation, 2-50
TABLO Input file, 3-70
TABLO-generated simulation, 2-36
Substituting variables
AnalyseGE, 3-105
Substitution, 3-102, 3-105
Subtotal, 1-9, 2-53
SUM
TABLO, 3-74
SUMHAR, 3-83
Systematic sensitivity analysis, 1-6, 1-10

—T—

Tableau
Equations matrix C, 2-55
TABLO
How to run, 2-35, 2-38
Record, Table files, 3-106
Restarting, 3-106
Stages, 3-106
TABLO Input file, 2-17, 3-61, 3-67
Comment, 3-73
Data manipulation, 3-82
Levels,Mixed,Linear, 3-100
Statements, 3-73
TABLO language, 3-76
TABLO stages, 3-102
TABLO Syntax
Example, 3-67
TABLO-generated program
Meaning, 2-20
TABmate, 1-12
Target shock, 1-14
Terminal input
Comments, 4-110
Terminal output
Suppressing, 4-113
Text data files, 3-83
Text files, 3-83
Input to MODHAR, 3-78
Use in GEMPACK, 4-119
Trailing spaces in file names
Not allowed, 4-121
Transfer statements, 1-13
Transferring data
Between PCs and mainframes, 3-84

—U—

Unlimited Executable-image version, 1-4
Unlimited Executable-image version licences, 1-4
UPDATE
Example, 3-93
Meaning, 3-93
Product, 3-93
Purpose, 3-86
UPDATE(CHANGE), 3-93
Updated data, 2-47
After Johansen simulation, 2-53
Updated data file, 4-120
User input
GEMPIE Example, 2-37, 2-39
User Input
Condensation in TABLO, 3-103
MODHAR Modify data, 3-81
TABL condense and Code, 3-107
TABLO - Step 1, 2-35
TABLO Check only, 3-107

—V—

VARIABLE
TABLO, 3-73
Variables, 2-55
Change, 3-75
Levels, 2-18
Percentage change, 2-18, 3-75
Vector variable
Meaning, 2-25
Vector variables, 2-55
Verbal description, 2-46
Versions of GEMPACK, 1-3
ViewHAR, 1-12, 3-83
ViewSOL, 1-12

—W—

Wilcoxen, Peter, 1-16
Windows 3.1

No longer supported, 1-12
WinGEM, 1-6
World Wide Web, 1-15
WRITE
TABLO, 3-74

—X—

XTransfer statements, 1-13