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

Fourth edition
October 1998

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

Fourth edition
October 1998

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, Clayton Vic 3168, 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; first, it enables the modeller to solve very large systems of non-linear equations; second, once the model equations have been specified using an algebra-like notation, the modeller is freed from the computing details of the solution process. The GEMPACK software calculates accurate solutions of the economic model.

GEMPACK runs on a wide variety of computers including

- 80386/80486/pentium computers running DOS, Windows or OS/2
- Unix machines
- DEC Vax and Alpha machines running VMS, 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 simulations. A recent development is the writing of Windows versions of some utility programs, easy-to-use in a Windows environment..

GEMPACK is fully documented with plenty of examples, including a range of example models.

This document is a practical introduction to GEMPACK. In particular it describes how to carry out simulations and how to build and/or modify models. 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.

The document also contains a guide to the rest of the GEMPACK documentation.

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)

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 Large-simulations Licence 1-4

1.2.6 WinGEM - The Windows Interface to GEMPACK 1-5

1.2.7 RunGEM - A Windows Program for Simulations with Different Models 1-5

1.3 Models Supplied with GEMPACK 1-6

1.4 Documentation of GEMPACK 1-7

1.5 A Guide to This Document 1-7

1.5.1 For New GEMPACK Users 1-7

1.5.2 Changes From Earlier Versions of GEMPACK 1-8

1.6 Contacting the Impact Project 1-9

1.7 GEMPACK World-Wide Web Site 1-9

1.8 GEMPACK-L Mailing List 1-9

1.9 Acknowledgments 1-10

2. HOW TO CARRY OUT SIMULATIONS WITH MODELS 2-11

2.1 The Steps in Carrying Out a Simulation 2-13

2.1.1 Steps using a TABLO-generated Program 2-14

2.1.2 Steps using GEMSIM 2-16

2.2 An Example Simulation with Stylized Johansen 2-18

2.2.1 Introduction to the Stylized Johansen Model 2-18

2.2.2 The Simulation 2-19

2.3 Installing GEMPACK Software 2-20

2.4 Using GEMPACK: WinGEM or Unix/Command Prompt 2-20

2.5 WinGEM : Stylized Johansen Example Simulation **2-21**
2.5.1 Starting WinGEM 2-21
2.5.2 Preparing a Directory for Model SJ 2-21
2.5.3 Setting the working directory 2-22
2.5.4 TABLO-generated Program or GEMSIM? 2-22
2.5.5 The Example Simulation using a TABLO-generated Program 2-23
2.5.6 The Example Simulation using GEMSIM 2-26
2.5.7 GEMSIM or TABLO-generated program? 2-28

2.6 Unix/Command prompt : Stylized Johansen Simulation **2-29**
2.6.1 The Example Simulation using a TABLO-generated Program 2-29
2.6.2 The Example Simulation using GEMSIM 2-32
2.6.3 GEMSIM or TABLO-generated program? 2-33

2.7 Interpreting the Results **2-34**

2.8 Specifying a Simulation **2-37**
2.8.1 Specifying a Simulation via a GEMPACK Command file 2-37

2.9 The Updated Data - Another Result of the Simulation **2-41**

2.10 How to Carry out Several Simulations At Once **2-42**
2.10.1 Preparing an Equations file for use by SAGEM 2-42
2.10.2 Running several Johansen simulations using SAGEM 2-43
2.10.3 The Results of These Johansen Simulations 2-45
2.10.4 The Command File SJLBJ.CMF for these Johansen simulations 2-47
2.10.5 SAGEM - Advantages and Disadvantages 2-48

2.11 How Johansen and Multi-step Solutions are Calculated **2-49**
2.11.1 The Linearized Equations of a Model 2-49
2.11.2 Johansen Solutions 2-50
2.11.3 Multi-step Simulations and Accurate Solutions of Nonlinear Equations 2-51

3. BUILDING OR MODIFYING MODELS 3-55

3.1 Writing Down the Equations of a Model **3-57**
3.1.1 Writing Down the Equations of Stylized Johansen 3-57

3.2 Data Requirements for the Linearized Equations **3-60**
3.2.1 Data Requirements for Stylized Johansen 3-60

3.3 Constructing the TABLO Input File for a Model **3-61**
3.3.1 Constructing Part of the TABLO Input File for Stylized Johansen 3-61

3.3.2 The TABLO Input File for the Stylized Johansen Model 3-64
3.3.3 Completing the TABLO Input File for Stylized Johansen 3-67
3.3.4 Change or Percentage-Change Variables 3-69
3.3.5 TABLO Language - Syntax and Semantics 3-70

3.4 Constructing Data Files 3-71
3.4.1 Constructing the Data File for Stylized Johansen 3-71
3.4.2 Modifying Data Using MODHAR 3-75
3.4.3 Using TABLO for Data Manipulation 3-76
3.4.4 Header Array and Text Data Files 3-77

3.5 Linearized TABLO Input Files 3-80
3.5.1 A Linearized TABLO Input File for Stylized Johansen 3-81
3.5.2 Noteworthy Features in the Linearized TABLO Input File 3-84
3.5.3 Writing UPDATE Statements 3-87

3.6 Levels TABLO Input Files 3-89
3.6.1 Levels TABLO Input File for Stylized Johansen 3-89

3.7 Mixed, Linearized or Levels TABLO Input Files? 3-93

3.8 TABLO Linearizes Levels Equations Automatically 3-94

3.9 Condensing Models 3-95
3.9.1 Substituting Out Variables 3-95
3.9.2 Backsolving for Variables 3-97
3.9.3 Omitting Variables 3-98
3.9.4 Stopping and Restarting TABLO 3-99

4. GEMSIM AND TABLO-GENERATED PROGRAMS 4-101

4.1 Actions in GEMSIM and TABLO-generated Programs 4-101

4.2 How These Programs Carry Out Multi-step Simulations 4-103
4.2.1 Results of a 4-step Simulation Looked at in Detail 4-105

4.3 Carrying out Simulations on Other Machines 4-107

4.4 Memory Required by TABLO-generated Programs 4-108

5. COMMON FEATURES OF GEMPACK PROGRAMS 5-109

5.1 Responding to Prompts 5-109
5.1.1 Default Response to Questions and Prompts 5-109

5.2 Comments in Input from the Terminal **5-109**

5.3 Interactive and Batch Operation, Stored-input and Log Files **5-110**

5.3.1 Invalid Input When Using Options 'sif' or 'asi' 5-112

5.3.2 Differences Between Batch and Interactive Program Dialogues 5-112

5.3.3 Terminal Output and Log Files 5-112

5.4 Other Program Options **5-113**

5.5 Increasing Parameters **5-114**

5.6 Error Messages **5-115**

5.7 Eliminating Syntax Errors in GEMPACK Command Files **5-116**

5.8 GEMPACK File Types, Names and Suffixes **5-117**

5.8.1 Files with System-determined Suffixes 5-117

5.8.2 Suffixes of Other Files 5-117

5.8.3 Files - Binary or Text? 5-118

5.8.4 Why So Many Files? 5-119

6. CHOOSING THE CLOSURE, SHOCKS, AND OTHER SETS OF VARIABLES6-121

6.1 Specifying the Closure **6-121**

6.1.1 Miniature ORANI Model 6-122

6.1.2 Specifying the Closure via a Command File 6-123

6.1.3 Component Numbers for a Variable 6-125

6.2 Choosing Other Sets of Variables **6-127**

6.2.1 Choosing Sets of Variables via a Command File 6-127

6.2.2 Choosing Sets of Variables Interactively 6-131

6.2.3 Simple Choices 6-132

6.2.4 Choosing Using the Lists Option 6-132

6.3 Specifying the Shocks **6-134**

6.3.1 Specifying the Shocks via a Command File 6-134

6.3.2 Specifying the Shocks Interactively 6-135

6.3.3 Shock Files 6-136

7. MORE ABOUT SIMULATIONS AND SOLUTION FILES 7-137

7.1 Several Simultaneous Johansen Simulations via SAGEM **7-137**

7.2 Printing the Results of a Simulation **7-139**

7.3 Subtotals **7-139**
7.3.1 Subtotals - An Example 7-139
7.3.2 Subtotals and Sets of Shocks in General 7-140
7.3.3 Storing Subtotals on Solution Files When Running SAGEM 7-140
7.3.4 Printing Subtotals or Individual Column Results using GEMPIE 7-141

7.4 Solution Files from Multi-Step Simulations **7-142**

7.5 How SAGEM and TABLO-generated Programs Solve Linear Equations **7-143**

7.6 Special Options for SAGEM and GEMPIE **7-144**
7.6.1 SAGEM Options 7-144
7.6.2 GEMPIE Options 7-145

7.7 Model Name, Version and Identifier **7-146**

8. THE OTHER GEMPACK PROGRAMS 8-147

8.1 Information from Equations Files **8-147**
8.1.1 Map of an Equations File Produced via SUMEQ 8-148
8.1.2 Homogeneity Simulations using SUMEQ 8-149

8.2 Looking at or Comparing Data on Header Array Files **8-150**
8.2.1 SUMHAR Summarizing a Header Array File 8-150
8.2.2 SEEHAR Prepare a Print File or CSV Spreadsheet File 8-150
8.2.3 CMPHAR Comparing Data on Header Array Files 8-151

8.3 Processing and Reporting Simulation Results using SLTOHT **8-153**
8.3.1 Header Array file output 8-154
8.3.2 Text file output, including spreadsheet output 8-156
8.3.3 Description File 8-158

8.4 Transferring Models Between Machines **8-160**
8.4.1 Transferring the Whole Model 8-161
8.4.2 Transferring Johansen Simulation Capability 8-161
8.4.3 Transferring Solution Files 8-163
8.4.4 Handling Large Data Files 8-163

8.5 Comparing Results from Different Machines **8-164**

9. A GUIDE TO THE REST OF THE DOCUMENTATION 9-166

9.1 Other Relevant Publications **9-166**

10. REFERENCES **10-168**

11. GEMPACK DOCUMENTS **11-170**

CHAPTER 1

1. Introduction

This document is an introduction to the current version (Release 6.0, October 1998) 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 data base 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[1] including

- 80386/80486/pentium microcomputers running DOS, Windows, Windows 95 or NT or OS/2,
- Unix machines,
- DEC VAX and Alpha machines running VMS, and .
- other mainframe, mini and microcomputers with an ANSI standard Fortran 77 or Fortran 90 compiler.

This document is a practical introduction to 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 gives more information about the programs used to carry out simulations, namely GEMSIM and the programs written by TABLO. Chapter 5 describes common features of the different GEMPACK programs. Chapter 6 documents the different ways sets of variables can be chosen (such as the set of variables to shock or to print). Chapter 7 gives more information about simulations (especially Johansen simulations) while chapter 8 is a guide to the utility programs, that is those not directly

---

[1] In this document, when we say "different machines" or "differ from machine to machine", we really mean different operating systems (such as those found on a Windows PC and a computer using Unix). GEMPACK programs run essentially unchanged on these different machines. However the commands to start them running usually differ between machines.

involved with simulations or building models. Chapter 9 is a guide to the rest of the GEMPACK documentation.

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 9 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.

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	for data management
SLTOHT	for post-solution processing of simulation results
SEENV	for determining the closure from Environment or Solution files

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

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

4. Windows programs (which are only available for 80386/80486/pentium PCs running Windows or Windows 95 or Windows NT):

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

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 80386/80486/pentium PCs running Windows 3.1, Windows 95 or Windows NT (or DOS or OS/2)[2]. 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 a Large-simulations licence.

For more information, contact us at the 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, these 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 it is installed. With the source-code version, a suitable Fortran compiler is required. Large models are usually solved using TABLO-generated programs, which (being model-specific) solve large models considerably faster than GEMSIM (which is general-purpose) can.

Source-code versions are currently available for 80386/80486/pentium PCs running Windows 3.1, Windows 95 or Windows NT (or DOS or OS/2), VMS (VAX and DEC Alpha) and 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.

This version of GEMPACK is similar to the version called the Executable-image version for Release 5.2 of GEMPACK. 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 26-sector MONASH) and for 10-region, 10-commodity GTAP; but it will not solve 35-sector MONASH, 50 -sector ORANIG or 10-region, 12-commodity GTAP. The full details of size limits for simulations with the Limited Executable-image version can be seen in section 6.2 of GPD-7.

---

[2] We expect that this version of GEMPACK will also run under Windows 98.

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

The Unlimited Executable-image Version and licence are new for Release 6.0. This 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 it 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 section 4 of GPD-8.]

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

1.2.4 Demonstration Version

This 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

1.2.5 Large-simulations Licence

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.[3] TABLO-generated programs produced with Release 6.0 of GEMPACK now require a **Large-simulations licence** 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 not require a licence to solve a 10-region,10-commodity version of GTAP but would require a Large-simulations licence to use the same program to solve most (or all) simulations with a 10-region,12-commodity version of GTAP.

[3] 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.

1.2.6 WinGEM - The Windows Interface to GEMPACK

On 80386/80486/pentium PCs running under Microsoft Windows or Windows 95 or Windows NT, 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 above). 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 chapter 9 of GPD-4 for more information about WinGEM.

1.2.7 RunGEM - A Windows Program for Simulations with Different Models

This is a Windows program (new for Release 6.0 of GEMPACK) 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).

See chapter 9 of GPD-4 for more information about RunGEM.

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 section 1 of GPD-8.

GEMPACK software makes it easy to transfer models between different computers (including different operating systems), as described in section 8.4 below, so you can easily exchange models with other modellers using GEMPACK.

1.4 Documentation of GEMPACK

GEMPACK is fully documented for users. The rest of the GEMPACK documents are described in chapter 9. You should be familiar with this document GPD-1[4], especially chapters 2-6, before reading the other documents.

Adobe Acrobat PDF versions of the GEMPACK documentation are available 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 5.2 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.

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.

Chapter 4 gives more information about carrying out simulations. When you want to know more about running the GEMPACK programs, read chapter 5. These give detailed suggestions for more efficient use of the programs whether you are running them interactively or in batch mode.

Chapter 6 tells you about various ways of specifying closures (exogenous/endogenous splits) of models, and of choosing sets of variables (such as the set of variables to shock or the set to print out results for). Read the relevant parts of this when you are ready to carry out simulations with your own model, or to change the closure and/or shocks for an existing model.

Read chapter 7 when you want to find out more about carrying out Johansen simulations. This tells you how to use the software to obtain several different (approximate) solutions very quickly, which is the great advantage of Johansen simulations.

Chapter 8 documents the utility programs, including the program SLTOHT which may be helpful in preparing reports based on your simulations. Read the relevant parts of this chapter when you need to

---

[4] 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".

find out more detailed information about a model (perhaps to understand your simulation results in more detail) or when you want to transfer your model to a different machine.

Chapter 9 is a guide to the rest of the GEMPACK documentation.

Getting Started

- If you are using an 80386/80486/pentium PC running Microsoft Windows 3.1 or Windows 95 or Windows NT, we encourage you to start working with WinGEM, the Windows version of GEMPACK. A hands-on introduction to GEMPACK using WinGEM is given in GPD-8, which we encourage you to work through.
- We encourage users on other machines to work through the relevant parts 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 From Earlier Versions of GEMPACK

This is addressed to readers who have used Release 5.2 or Release 5.1 of GEMPACK.

The new features in Release 6.0, and the changes from Releases 5.2 and 5.1, are documented in GEMPACK document GPD-4.

If you are on an 80386/80486/pentium PC running Windows 3.1, Windows 95 or Windows NT, you will probably want to work with WinGEM, the Windows version of GEMPACK. If you are not familiar WinGEM, we encourage you to work through the hands-on introduction to WinGEM given in GPD-8. WinGEM does not change any of the modelling steps you are used to but rather provides a Windows interface to essentially the same GEMPACK programs running in DOS boxes. However we think it will make modelling easier even for experienced DOS and GEMPACK users.

1.6 Contacting the Impact Project

For more information about GEMPACK, contact

The GEMPACK Manager
Centre of Policy Studies and Impact Project
Monash University
Clayton, Vic 3168
Australia
email: impact@vaxc.cc.monash.edu.au
telephone: (03)-9905-5112 or +61-3-9905-5112 from overseas
FAX: (03)-9905-5486 or +61-3-9905-5486 from overseas

The GEMPACK developers can be contacted as above.

1.7 GEMPACK World-Wide Web Site

The GEMPACK World-Wide Web site is at address

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

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

The first edition (1993) of this document benefited from our reading of Wayne Calder's "Hitchhiker's Guide to GEMPACK" (Calder 1992), which is an introduction to Release 4.2.02 of GEMPACK; we are grateful to Wayne for making a preliminary copy available to us while we were preparing the first edition of this document. We are also grateful to Paul Gallagher and Colin Thompson for assistance with, and feedback about, the first edition of this document.

We are grateful to Peter Wilcoxen for suggesting the use of Command files (see, for example, section 2.8.1 and chapter 19 of GPD-4) 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 and ViewSOL 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 those who beta tested Release 5.0, Release 5.1, Release 5.2 and/or Release 6.0.

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 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 tonne) and its quantity Q (tonnes). 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 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 a so-called **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 so-called **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 so-called **Solution file**.

Step 3. Printing 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).

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.)

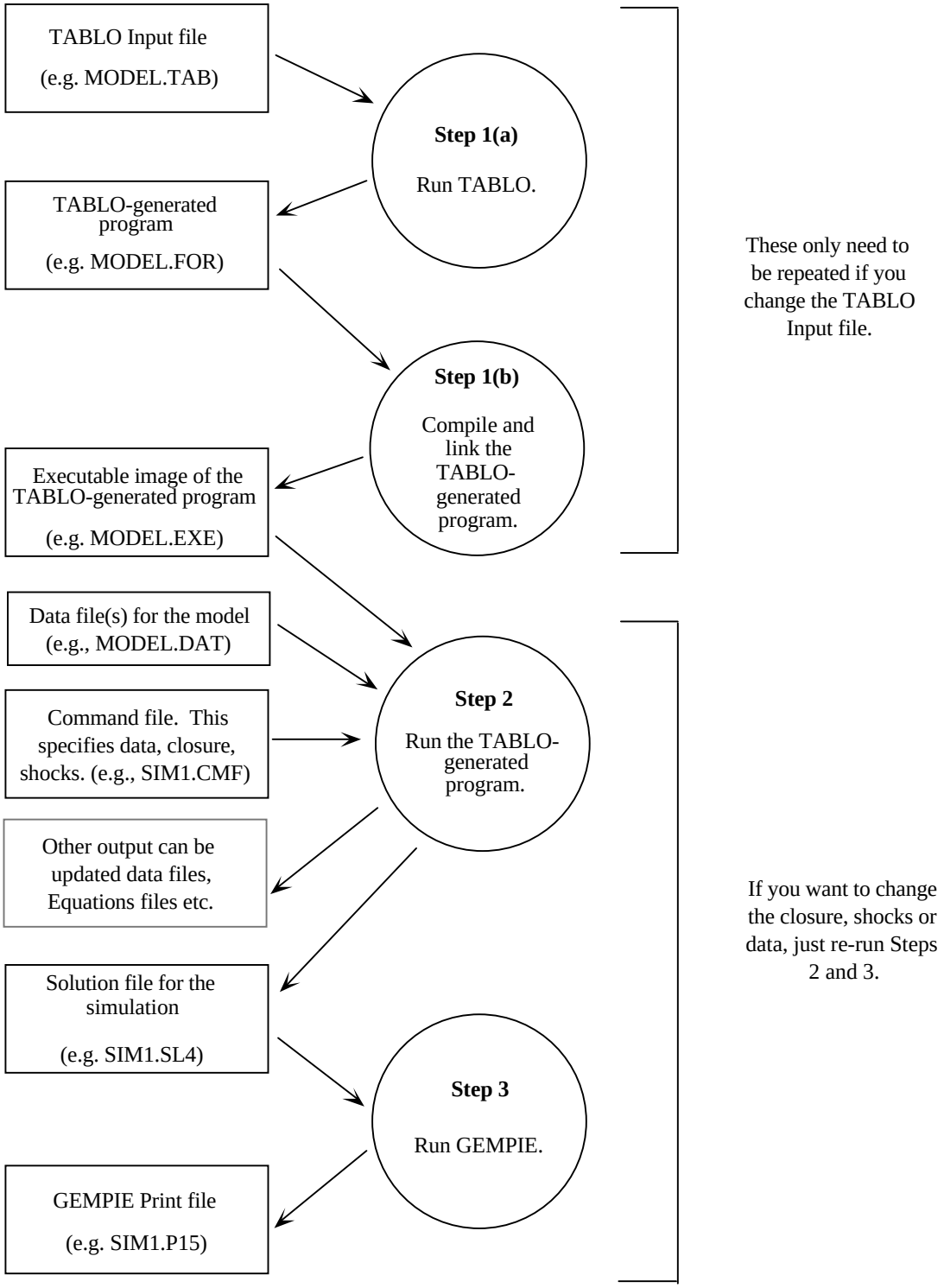


Figure 2.1.1 The Steps in Carrying Out a Simulation using a TABLO-generated Program

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 so-called **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 [5] and tell it to use the GEMSIM Auxiliary files produced in Step 1. Take inputs from a so-called **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 so-called **Solution file**.

Step 3. Printing the Results of the Simulation

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

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.

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

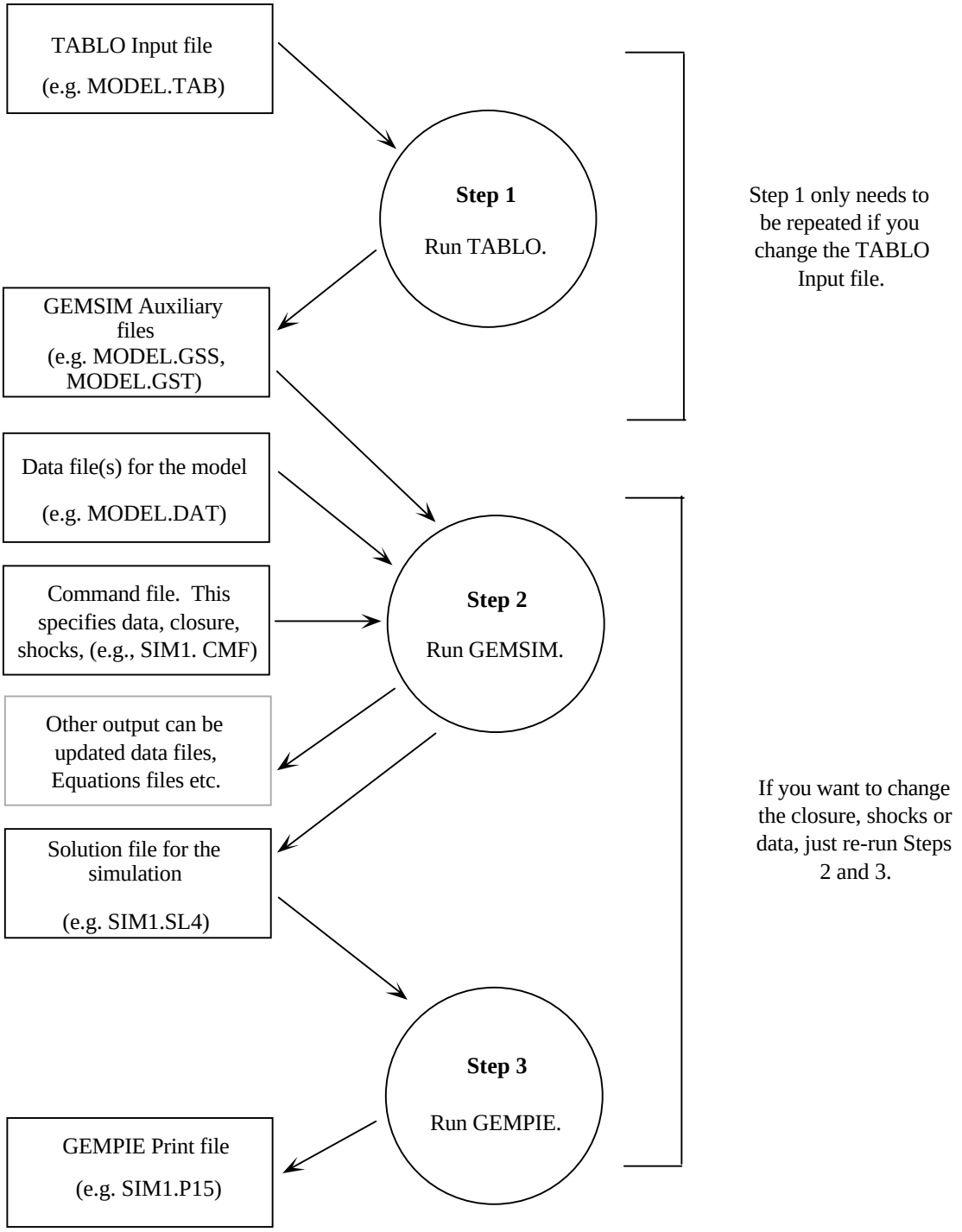


Figure 2.1.2 The Steps in Carrying Out a Simulation using GEMSIM

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 2 sectors "s1" and "s2" each producing a single commodity, one household sector and 2 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 Sectors	1	4.0	2.0	2.0	8.0
Commodity	2	2.0	6.0	4.0	12.0
Labor Factors	3	1.0	3.0		4.0
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** [6]
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 value of inputs of commodity i to industry j	(none)
DVFACIN(f,j)	Dollar value of usage of factor f to industry j	(none)
DVHOUS(i)	Dollar value of household use of commodity i	(none)

Table 2.2.1b: Levels Variables of Stylized Johansen

[6] In this column, a colon ':' indicates a subscript in DPPW, so that, for example, P:i means P with a subscript i. "(none)" means that this variable has no explicit counterpart in DPPW.

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 DOS machine, 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 using Windows on your PC (Windows 3.1, Windows 95 or Windows NT), you should also install the Windows programs

- the Windows programs, **WinGEM** (the Windows interface to GEMPACK), and
- **ViewHAR** (the Windows viewer for Header Array data files)
- **ViewSOL** (the Windows Solution viewer).

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, OS/2 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).

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.

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.]



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 File Manager in Windows 3.1 or Explorer in Windows 95 or NT, or by using the DOS *copy* command.

(a) To copy these files within Windows,

change to File Manager in Windows or Explorer in Windows 95 by the usual Windows method. Use File Manager or 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 F77L3 or LF90) 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

TABLO Implement
Compile & Link
Run TG Program
GEMSIM Solve
SAGEM Johansen Solve
GEMPIE Print

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

TABLO Implement
Compile & Link
Run TG Program

GEMPIE Print

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

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.

[7] 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.

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 is the name of the Solution file?

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***.[8]

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.

---

[8] 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[9]

TABLO Implement
Compile & Link
Run TG Program
GEMSIM Solve
SAGEM Johansen Solve
GEMPIE Print

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

TABLO Implement

GEMSIM Solve

GEMPIE Print

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.

[9] 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**.[10] This is all TABLO needs to implement the model.

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 is the name of the Solution file?

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

---

[10] 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.

[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***.[11]

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 section 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.]

---

[11] 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**),[12]
- 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 [13]

```
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

```
solution file = sjlb ;
```

---

[12] 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.

[13] 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** the name of the Solution file to create during the run you have just completed. (This Solution file contains the solution of the simulation.) 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.cmf** has been created. 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.]

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 producung 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 section 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.)

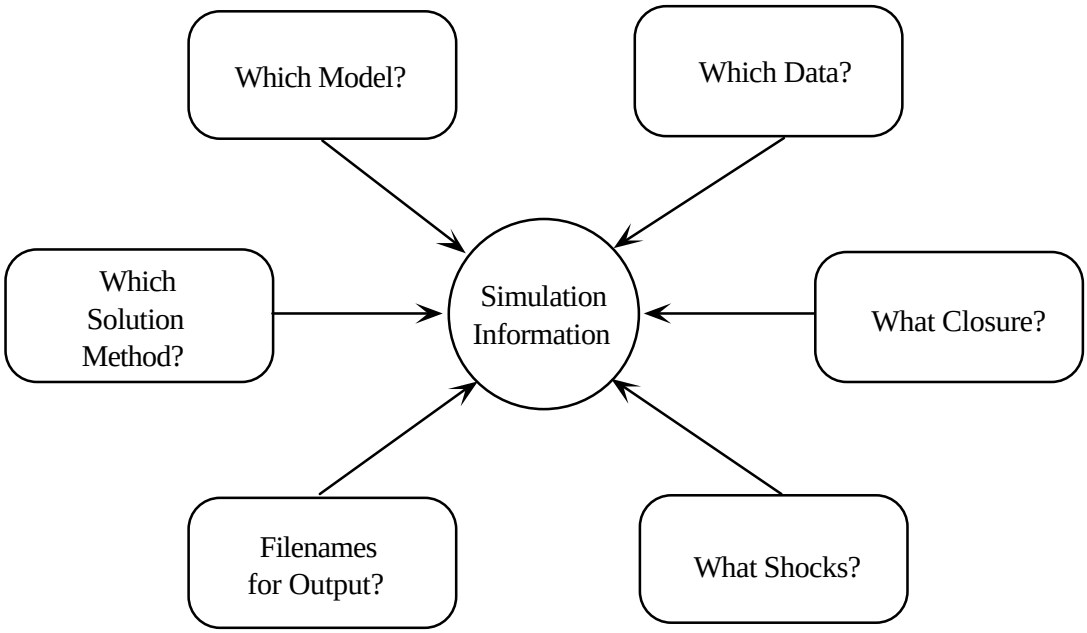


Figure 2.8 The Information Required to Specify a Simulation

Within GEMPACK, the normal way of specifying this information to the software is via a so-called **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 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.]

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 for model
auxiliary files = sj ;

! Data files
file iodata = sj.dat ;
updated file iodata = sjlb.upd ;

! Closure
exogenous p_xfac ;
rest endogenous ;

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

! Simulation part
solution file = sjlb ;
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).

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.

The statement

```
solution file = sjlb ;
```

tells SJ.EXE or GEMSIM the name of the Solution file to contain the solution of the simulation. (The suffix '.SL4' is added automatically by the program, so that the Solution file is called SJLB.SL4.)

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 = sjlb.upd ;
```

names the file to contain this updated data. (The name includes 'LB' to remind us that this data depends on the labor shock.) [You will get an opportunity to examine the updated data for this simulation in section 2.9 below.]

You are required to give a so-called **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 a so-called **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.

Complete documentation of the statements that can be used in a GEMPACK Command file for running TABLO-generated programs or GEMSIM is given in chapter 19 of the GEMPACK document GPD-4.

2.9 The Updated Data - Another Result of the Simulation

As you saw in the section above, the line

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

in the Command file SJLB.CMF means that, when you ran the simulation, the software produced the so-called **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 1.1 and 1.2 of GPD-8[14] 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 Sectors	1	4.235	2.118	2.118	8.471
Commodity	2	2.118	6.353	4.235	12.707
Labor Factors	3	1.059	3.177		4.235
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.

[14] These examples are for WinGEM users. Command prompt users can look at Example 8 in section 3 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**[15]**,** 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.

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-

[15] 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.[16]

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.

---

[16] 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[17] more quickly (that is, using less computing resources) than one single, more accurate, multi-step simulation result.

[17] 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 chapter 19 in GPD-4.)

When you run SAGEM, the Solution file produced can contain several individual column results and/or cumulative results [18] and subtotals. More information about these is given in chapter 7 below.

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.

[18] 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[19] 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

$$Cz = 0 \qquad (1)$$

where

C is the n x m matrix of coefficients [20] of the equations,
known as the Equations Matrix,

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[21] 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.

---

[19] 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.

[20] 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.

[21] 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 $Cz = 0$ 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.[22]

[22] 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 MA28 to solve (3). These solve (3) by calculating a so-called 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 7.5 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.

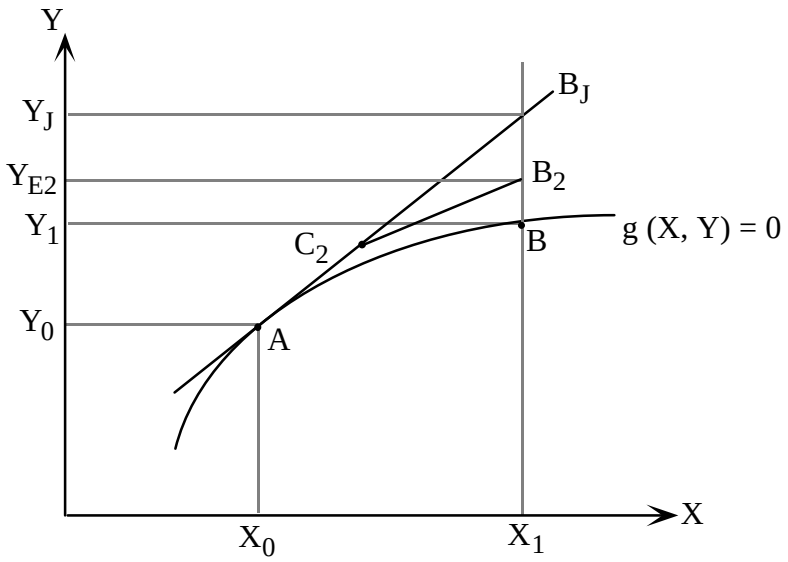


Figure 2.11.3 Illustration of Euler's Method

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 5.2 of GPD-2. 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 [23]		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.)

[23] 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.

Readers who have used Release 4.2.02 of GEMPACK will know that this always doubled the number of steps from solution to solution when extrapolating, but this is not necessary. The only restriction 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 5.2 of GPD-2 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.

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.[24] 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 the last column of this table, we use a colon **:** to indicate subscript, as in P:i which means P with subscript i.) In DPPW

[24] 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.

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.

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[25]

```
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,

[25] 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[26]

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

---

[26] 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.

---

[27] 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[28]

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

[28] 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 2.2 of GPD-4.

```
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 so-called **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** (User's Guide to TABLO, GEMSIM and TABLO-generated Programs), 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 text files. In the latter case you can create them using your favourite editor. The syntax required for text files is introduced in the next subsection, section 3.4.1, and is explained in detail in chapter 20 of GPD-4.

While text 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 utility programs SUMHAR and SEEHAR enable you to examine the data on these files.

The routine way of creating a Header Array data file is to first create one (or more) text 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

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 chapter 20 and section 5.3.4 of GPD-4.)

The actual data is, by default, in ROW order, with each row starting on a new line of the file.[29]

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
!
```

[29] 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 20 of GPD-4.
Users familiar with Release 4.2.02 of GEMPACK will note that the format of arrays on a text file is considerably different from that required by that release of the software. However, it is quite easy to convert text files used with Release 4.2.02 to be used with Release 5 or later - see chapter 20 of GPD-4 for details. Note also that Header Array files prepared with an earlier release of GEMPACK require no changes in order to be compatible with Release 5 or later.

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.[30]

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[31]
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.[32] 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 5.3.2 below.)

[30] A new feature of Release 5.1 of GEMPACK is 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 20 of GPD-4 for details.

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

[32] 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, which are described in 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.[33]

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.[34] 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.[35]

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

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

[35] If you wish, you can follow the method in Example 1.1 or 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).[36]

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 GPD-3.

[36] 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 section 8.4 of GPD-2 for a brief discussion of this.

See section 4.12 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.

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 8, MODHAR is documented in GPD-3, and ViewHAR (which is only available on Windows PCs) is introduced in section 2 of GPD-8.

Text 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 20 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 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 Kermit or 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 section 8.3 below and in chapter 6 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 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 of GPD-8); you might like to look at its text data file [37] 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 4.6.1 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 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 so-called File "qualifiers" (see sections 3.5 and 4.6 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.

[37] 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 format required for Release 4.2.02 of GEMPACK. (See section 20.3 of GPD-4 for more details about these changes.)

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-3
Header Array file	Text file(CSV) (row order) (column order)	SEEHAR	SS ROW COL	8.2.2
Solution file	Header array file	SLTOHT	-	8.3
Solution file	Text file(CSV) (row order) (column order)	SLTOHT	SS,SSS SIR SIC	8.3 and GPD-4, chapter 6
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**.[38]

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 so-called **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 A.2 of Appendix A in GPD-2.

[38] 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,[39] 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 #
```

[39] 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.[40]

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,

[40] 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. [41] 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

[41] 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 A.2 of Appendix A in 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.[42]

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 A.2 of Appendix A in 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 ;

[42] 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.[43]

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.

---

[43] 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                           !
!-----------------------------------------------------------------------!

!   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               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)
      # Total demand for (or supply of) commodity i #
                               ! This is x:i (i=1,2) in DPPW ! ;
VARIABLE (all,f,FAC)   XFAC(f)
      # Total demand for (or supply of) factor f    #
                               ! This is x:i (i=3,4) in DPPW ! ;
VARIABLE (all,i,SECT)  XH(i)   # Household demand for commodity i #
                               ! This is x:i0 (i=1,2) in DPPW ! ;
VARIABLE (all,i,SECT) (all,j,SECT)     XC(i,j)
      # Intermediate inputs of commodity i to industry j #
     ! This is x:ij (i,j=1,2) in DPPW ! ;
VARIABLE (all,f,FAC)(all,j,SECT)       XF(f,j)
      # Factor inputs to industry j #
     ! This is x:ij (i=3,4; j=1,2) in DPPW ! ;
VARIABLE (all,j,SECT)  W(j)    #Price expression#;
```

```
!-------------------------------------------------------------------!
!     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)   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                    !
!-------------------------------------------------------------------!

!   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 ;

!   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. Formulas for the parameters   !
!...................................!
! FORMULAs for ALPHAH(i) and Q(j) are not needed since they do not
     appear in the linearized equations produced by TABLO. !

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) ;

! Initial value of Y !
FORMULA  Y = SIM(i,SECT, PC(i)*XH(i) ) ;

!-------------------------------------------------------------!
!  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.

Another factor to keep in mind is the efficiency of the different implementations - that is, the speed with which they solve and the memory required to solve them. Our experience with ORANI-F, reported in section 5.3 of Harrison *et al* (1993a), found that the linearized implementation was more efficient than our mixed one. At this stage, we are not sure if this is true in general, and we hope to compare the efficiency of different implementations of other models. It is possible that mixed representations will become more efficient as modellers gain more experience in working with them.

A fuller discussion of this whole question can be found in Harrison *et al* (1993a).

Advice about linearizing equations by hand can be found in section A.2 of Appendix A in 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.[44]

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 sections 2.2.1 to 2.2.3 of GPD-2.

[44] 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 so-called **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 5.3.2 below), a warning message asking you to check that A8(i) cannot be zero is written to the terminal and to the Information file, and TABLO continues with the substitution.

If you proceed with the substitution and some value of A8 is indeed zero, the error will be detected when you run the TABLO-generated program.

In order to perform a substitution when running TABLO, you merely say which variable to substitute out and which equation to use; you can nominate the equation by giving its name or its number. Then TABLO automatically rewrites all the remaining EQUATIONs and UPDATEs.

If you substitute out a variable with k components, this reduces by k the number of rows and the number of columns of the matrix in the system Cz=0 of equations to be solved.

Because the variable substituted out no longer appears in any equations, it must be an endogenous variable. Also, you cannot see its values in any simulation, since it does not appear in the resulting Equations or Solution files.

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).[45]

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

[45] 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.

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.[46]

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 large models; for small models it is best to always start from the Check.

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).

---

[46] 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.[47]

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.

[47] 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.

CHAPTER 4

4. GEMSIM and TABLO-generated Programs

4.1 Actions in GEMSIM and TABLO-generated Programs

When you build a model in GEMPACK you write down the theory in the TABLO Input file for your model. When you run TABLO, you can choose either to produce output for GEMSIM (the GEMPACK Auxiliary files) or to produce the Fortran program which we call the TABLO-generated program for your model. When you run GEMSIM or this TABLO-generated program and attach the data, it can carry out various **actions** (see Figure 4.1).

If the TABLO Input file for your model contains either

- one or more levels EQUATIONs, and/or
- linearized equations and one or more UPDATE statements or one or more levels VARIABLEs,

the most important action is that of carrying out a (possibly multi-step) simulation with the model (the first case illustrated in Figure 4.1). This produces a Solution file containing simulation results (percent changes and possibly pre- and post-levels results). It also produces updated data files and other potentially useful files (including Equations and Environment files). For example, for the TABLO Input file for the Stylized Johansen Model given in section 3.3.2, the database on file "iodata" is updated to contain the new values after the simulation (see section 2.9).

Really, all models implemented with GEMPACK should fall into the category covered in the previous paragraph. This is because the underlying levels equations of an economic model are bound to be nonlinear so that multi-step calculations are required to solve them accurately. However there are still a few models implemented via TABLO Input files which contain linearized EQUATIONs but no levels EQUATIONs, levels VARIABLEs or UPDATE statements.[48] It is only possible to carry out Johansen simulations with such a model, and these must be carried out using SAGEM. The most important action for GEMSIM or the TABLO-generated program associated with such a model is that of creating the Equations file for the model (the second case illustrated in Figure 4.1). This Equations file is then the starting point from which you can carry out Johansen simulations by running the program SAGEM. Of course, in this case, when there are no UPDATEs either explicit or generated by levels variables, you do not get any updated data files. Note that, in this case, GEMSIM or the TABLO-generated program do not carry out the simulation but only create the Equations file. You must then run SAGEM to carry out a simulation.

In either of the two cases above, other possible actions are those of doing the DISPLAYs or WRITEs (if there are any) in your TABLO Input file.

Unless you directed otherwise (via the options menu at the start of the CODE stage of TABLO - see section 5.10 of GPD-2), GEMSIM and/or the TABLO-generated program are capable of carrying out all the actions in the TABLO Input file. When you run either GEMSIM or the TABLO-generated

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

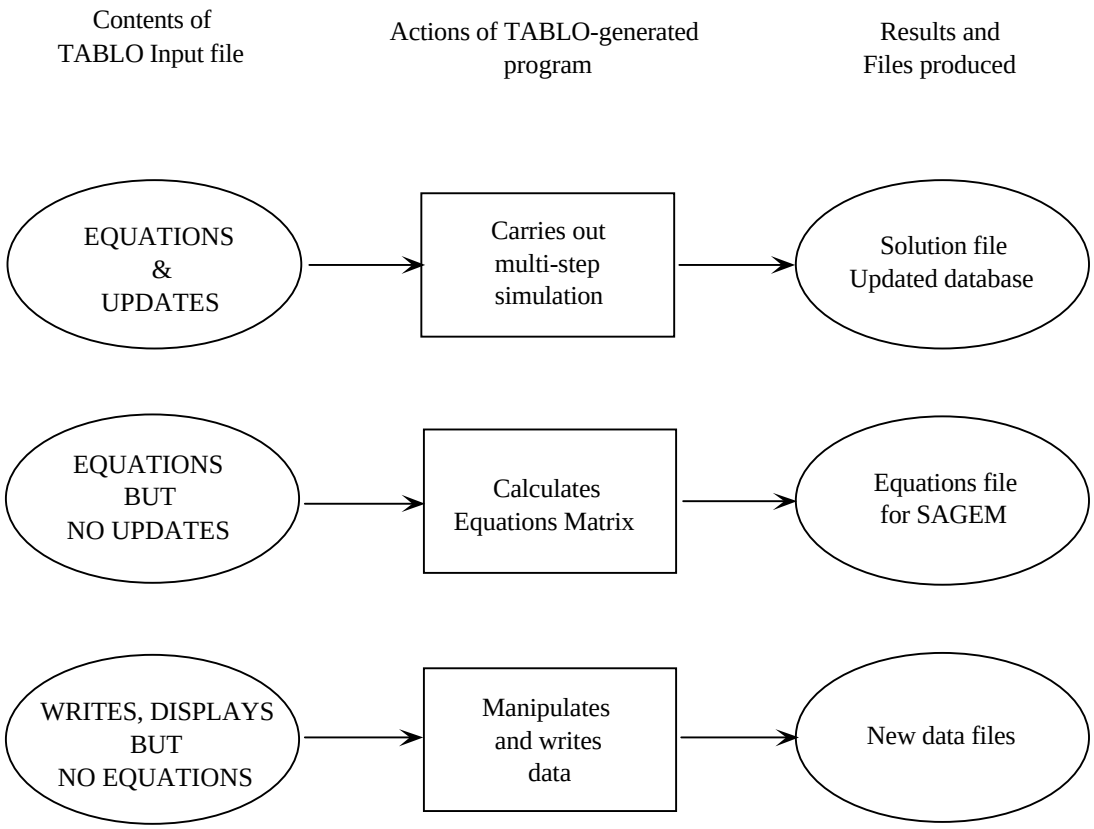


Figure 4.1: Actions of GEMSIM and TABLO-generated Programs

program, it will normally carry out all of these actions. However, by selecting one or more of the options from the menu presented when you start it running, you can choose to do only some of the possible actions. For example, you may wish to ignore the DISPLAY statements and just carry out a multi-step simulation. If so, select the option **NDS** when you run GEMSIM or the TABLO-generated program. Alternatively you can add the statement **NDS = yes ;** to your Command file.

Of course, equations are essential for a simulation and so, if you select the option **NEQ** ("Do no equations"), this means that no simulation will be carried out either, and the only actions possible are WRITEs or DISPLAYs.

Sometimes TABLO Input files are created for data manipulation rather than for modelling (see section 3.4.3 above for an example). In this case there are no EQUATIONs (and hence no VARIABLEs or UPDATE statements) and the only possible actions are DISPLAYs and WRITEs (the third case illustrated in Figure 4.1). The WRITEs are usually the main actions since they can produce new (or modified) data files. Again, if the program is capable of doing some WRITEs and some DISPLAYs, you can elect, in any one run of the program, to do just the WRITEs or just the DISPLAYs.

Note that TABLO-generated programs and GEMSIM are also usually able to carry out some “extra” actions (TABLO-like statements in the Command file rather than in the TABLO Input file) - see chapter 9 of GPD-4. Usually these “extra” actions are carried out during the reads/formulas part of step 1 (see Figure 4.2).

We describe in section 4.2 below how GEMSIM and TABLO-generated programs carry out multi-step simulations. In section 4.3 we tell you how you can carry out simulations on machines on which GEMPACK is not installed, and, in section 4.4, we say a little about the amount of memory required by

TABLO-generated programs. More information about GEMSIM and TABLO-generated programs can be found in chapter 5 of GPD-2.

4.2 How These Programs Carry Out Multi-step Simulations

Each step of a single multi-step simulation consists of five main parts:

1. sets, subsets, reads, writes, displays, calculations of formulas, and any “extra” statements,
2. calculation of equations (one submatrix at a time),
3. the simulation part,
4. backsolving (if required),
5. updating the data.

(See Figure 4.2 for these parts, some of which are shown there in a little more detail.)

For a Johansen simulation, each of the above is done just once. For a single N-step simulation, each is done N times if you are using Euler's method or the midpoint method while each is done (N+1) times if you are using Gragg's method.[49]

If you are extrapolating on the basis of two or more multi-step solutions, each solution is calculated (first the one with the smaller number of steps, then the one with the next most, and so on).

Finally, the extrapolation is carried out. This involves taking an appropriate linear combination of the different solutions (and updated data bases).

Normally DISPLAYs and WRITEs and any “extra” actions are only done during part (i) of the first step of the first multi-step solution, when values written or displayed reflect values in the original data. [If you are using existing Equations and Base Coefficient Values files, or an LU file, to start the simulation, note that 1 and 2 above are not done during step 1, so that DISPLAYs, WRITEs and “extra” actions are usually not done.]

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

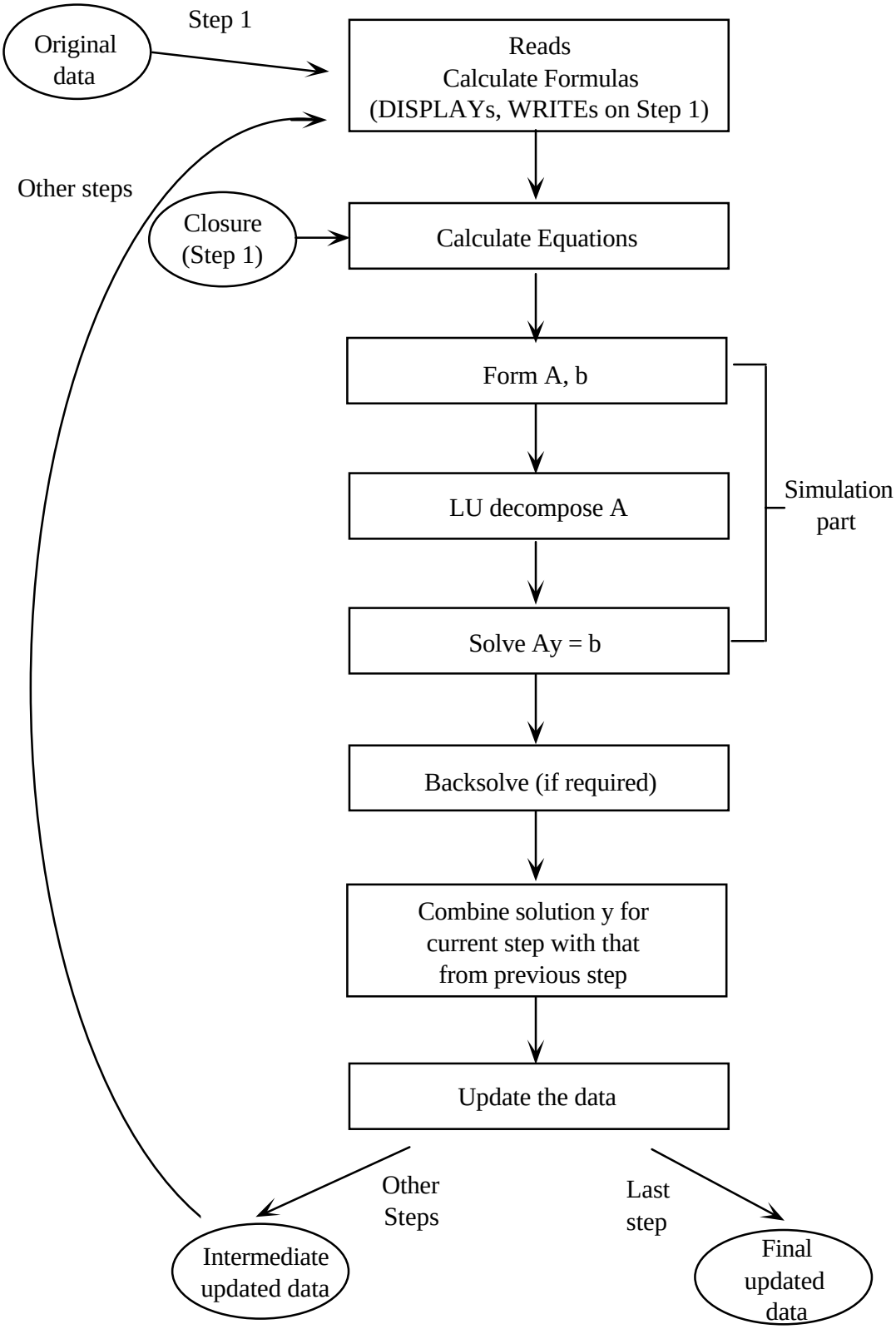


Figure 4.2: Calculation of One Multi-step Solution

In an N-step simulation, the shocks are broken into N equal parts. In step number K, the formulas and equations are calculated on the basis of the data as updated after step K-1 (if K is at least 2, or the original data if K=1), and the effect of applying just the part of the shock relevant to this step is calculated; this gives a solution, say y(K) for just these shocks. (See below for more explanation as to how y(K) is calculated.) This solution y(K) is added to the combined solution x(K-1) before this step to produce the combined solution x(K) after K steps. [x(K) reflects the movements in the endogenous variables corresponding to just K Nths of the total shocks.] Thus,

```
x(1) = y(1)      step 1
x(2) is obtained from x(1) and y(2)
x(3) is obtained from x(2) and y(3), and so on.
```

We give a concrete example in section 4.2.1 below.

At the end of each step the data is updated on the basis of the step solution y(K). Specifically, the solutions in y(K) are applied to the updated data after step K-1 to produce the updated data after step K. This data is the starting point for step K+1 (or is the final updated data at the end of this multi-step simulation if this is the last step).

At step K, the linear equations (based on the data at the start of this step) are formed up as

$$C(K).z = 0$$

much as in equation (1) of section 2.11.1 above. Note that, in a multi-step simulation, the Equations Matrix C changes from step to step reflecting the changes in the data as it is updated, which is why we use the notation C(K) above. Then, taking account of the closure and of the shocks for this step, the equations are expressed as

$$A(K).y(K) = b(K)$$

much as in equation (3) of section 2.11.2 above. Here the LHS matrix A(K) and the RHS vector b(K) depend on K as does the solution y(K) for this current step. These equations are solved as described in section 7.5 below: first the LU decomposition of A(K) is carried out (this is the time-consuming part) and then the values of y(K) are calculated. The solution y(K) is normally made as accurate as possible by using iterative refinement (see section 7.5 below).

Note that, if you condensed your model, some backsolving may be done even if you didn't explicitly ask for the values of any eliminated (endogenous) variables to be calculated. These are done if they appear to speed up the updates (which are done after backsolving).

4.2.1 Results of a 4-step Simulation Looked at in Detail

In this section we look at the results of the 4-step version of the simulation carried out in section 2.2 above. This is the simulation with Stylized Johansen in which the supply of labor is increased by 10%. We show how the shock for each of the steps is calculated, and show how the final results for certain endogenous variables are built up from the results of the individual steps.

Note that it is not necessary to understand this in order to carry out multi-step simulations with GEMPACK, so you may well prefer to skip this section at present.

The results discussed in this section are those obtained using Euler's method. (Gragg's method or the midpoint method would give different results.)

(a) How the Shock is Broken Up

Imagine that the initial supply of labor is 100 units; we show how the 10% shock to it is broken up into the shock for each of the 4 steps.[50]

Over the whole simulation, the labor supply must increase by 10%, that is, from 100 to 110, which is an overall increase of 10 units. The shocks at each step are always calculated to ensure that

the same increase (in levels) occurs at each step.

Thus, at each step, the supply must increase by 2.5 (=10/4) units.

In step 1, supply must increase from 100 to 102.5 which is an increase of (25/100)x100=2.5%.
In step 2, supply must increase from 102.5 (its value after step 1) by a further 2.5 units to 105 units; this is an increase of (25/102.5)x100=2.439%.
Similarly in step 3 the increase is (2.5/105)x100=2.381%
while in step 4 it is (2.5/107.5)x100=2.326%.

Step	Supply at start	Increase	Supply at end	% Increase
1	100	2.5	102.5	2.5
2	102.5	2.5	105	2.439
3	105	2.5	107.5	2.381
4	107.5	2.5	110	2.326

(b) Calculation of Results for Endogenous Variables

Consider first what happens to just one of the endogenous variables, namely p_XH("s1"), as these four partial shocks are applied to the model.

In the first step, the levels variable XH("s1") increases by 1.5%.
For simplicity (again this makes no difference to the final outcome) we suppose that the initial value of the Levels variable XH("s1") = 100 as well. After the first step, XH("s1") = 101.5 units.[51]

In the second step, in response to the 2.439% shock to XF("s1"), the increase in XH("s1") = 1.463%, the updated value of the levels variable XH("s1") is calculated from

XH("s1") = 101.5 * (1 + 1.463/100) = 102.985 units

an increase in the combined solution of 2.985%.

In the third step, in response to the 2.381% shock to XF("s1"), the increase in XH("s1") = 1.429%, and so the levels variable XH("s1") is updated to

XH("s1") = 102.985 * (1 + 1.429/100) =104.457

In the fourth step, in response to the 2.381% shock to XF("s1"), the increase in XH("s1") = 1.395%,

---

[50] The assumption about the initial supply being 100 is, in fact, irrelevant here. You can easily check this by replacing 100 by some other number (perhaps 4, which would be the supply for the data in Table 2.2.1a if the price is one dollar per unit).

[51] Note that, as you would expect, in the first step, the 1.5% increase in XH("s1") is exactly one quarter of what it would be in the corresponding 1-step simulation (namely 6.0%, as you can see by looking at the 1-step column in the Extrapolation Accuracy file SJLB.XAC discussed in section 2.11.3 above) since the shock is exactly one quarter.

and so the levels variable XH("s1") is updated to

XH("s1") = 104.457 * (1 + 1.395/100) =105.914

an overall increase in the four steps of 5.914%

Using the notation from section 4.2 above, namely letting y(K) denote the step solution at the Kth step, and x(K) the combined solution after K steps, the results for p_XH("s1") are summarised in the following table.

Step	y(K)	x(K)	
1	1.5	1.5	x(1) = y(1)
2	1.463	2.985	x(2) calculated from x(1) and y(2)
3	1.429	4.457	x(3) calculated from x(2) and y(3)
4	1.395	5.914	x(4) calculated from x(3) and y(4)

Thus the 4-step simulation result for the variable p_XH("s1") (in response to the 10% shock to XF("s1")) is 5.914%. This is the result for the 4-step Euler solution (not the extrapolated solution) on the Extrapolation Accuracy file SJLB.XAC, (the results are shown in section 2.11.3 above).

The table below shows the same information for the endogenous variable p_XH("s2").

Step	y(K)	x(K)	
1	1.75	1.75	x(1) = y(1)
2	1.707	3.487	x(2) calculated from x(1) and y(2)
3	1.667	5.212	x(3) calculated from x(2) and y(3)
4	1.628	6.925	x(4) calculated from x(3) and y(4)

Thus the 4-step Euler simulation for the variable p_XH("s2") (in response to the 10% shock to XF("s1")) is 6.925%.

4.3 Carrying out Simulations on Other Machines

Once you have built a model, you can move it and the software for carrying out simulations with it, to other machines with the same operating system but on which GEMPACK is not installed.

For a model solved using GEMSIM, this means that you must copy the executable image of GEMSIM and the model's Auxiliary Statement and Table files (these usually have suffixes '.GSS' and '.GST') to the other machine.

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

Of course you will also need to copy any data files used in the simulation, and any relevant Command files or Stored-input files.

You can also copy the executable image of GEMPIE to the other machine in order to produce human readable output from your simulation. However you cannot move other GEMPACK programs to a machine on which GEMPACK is not installed because these programs need to access the GEMPACK Licence file and to have other operating system features set up appropriately.

Note that, because the Auxiliary Statement and Table files are binary files, you cannot move TABLO-generated programs produced on one type of machine (for example, a Unix machine) directly to another type of machine (for example, a PC). Rather, to move a model from one type of machine to another on which GEMPACK is installed, follow the procedure described in section 8.4 below (which involves transferring the TABLO Input file and the data files).

4.4 Memory Required by TABLO-generated Programs

Memory constraints may become a problem on PCs. If so, TABLO has options (documented in section 5.10.2 of GPD-2) for reducing the amount of memory required by TABLO-generated programs, usually at the cost of increasing their execution time. Suggestions for dealing with memory constraints are given under the heading "Advice on Which of These to Select" near the end of section 5.10.2 of GPD-2.

There are not corresponding "low memory" options for GEMSIM.

CHAPTER 5

5. Common Features of GEMPACK Programs

Although the different GEMPACK programs are designed for carrying out different tasks, they all follow certain conventions and operate in similar ways, as explained in this chapter. For example, they process your responses to prompts in a consistent way (see section 5.1), they all allow comments starting with a single exclamation mark in input from the terminal (see section 5.2), and they all offer standard options (such as the ability to direct output to a Log file) as explained in sections 5.3 and 5.4. The procedure by which you increase the amount of memory available to a program is explained in section 5.5 while section 5.6 describes the GEMPACK error reporting scheme. Section 5.7 gives advice about identifying and correcting syntax errors in GEMPACK Command files. Section 5.8 is about file names, file suffixes and GEMPACK file types.

5.1 Responding to Prompts

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.)

5.1.1 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.

5.2 Comments in Input from the Terminal

In processing terminal input, all GEMPACK programs ignore any part of an input line starting with a single exclamation mark '!'. Since input on a Stored-input file is treated as if it were input from the terminal, this also applies to Stored-input files. 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.

5.3 Interactive and Batch Operation, Stored-input and Log Files

By default all GEMPACK programs operate in an interactive mode in which the program prompts you for information and you respond by typing at the terminal. If your input is judged by the program to be invalid, the program offers an explanation and then gives you a chance to input a different response.

This is most appropriate when the program in question requires only a relatively small amount of user input. When large amounts are required, responding at the terminal can be time-consuming and error prone. Equally importantly, it can then be difficult (perhaps impossible) to reproduce your results or be sure after the event as to exactly what your results mean. For these reasons it is often important to be able to run the programs by first preparing (on a Stored-input file) all the input required and then using this file to run the program. We refer to this as running the program in **batch mode**.

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'.)

GEMPACK contains a facility (independent of the operating system) whereby any of its programs can be instructed to take their input from a Stored-input file. This is done via the options which are presented at the start of the run of every GEMPACK program. When you start a program running you can select one of several options including the following:

```
      ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        GEMPACK OPTIONS
                  ( --> indicates those in effect )

     BAT Run in batch          STI Take inputs from a Stored-input file
     BPR Brief prompts         SIF Store inputs on a file
     LOG Output to log file    ASI Add to incomplete Stored-input file

Select an option   :  <opt>  Deselect an option      : -<opt>
Help for an option : ?<opt>  Help on all options     : ??
Redisplay options  : /       Finish option selection : Carriage return

Your selection >
```

Options Screen for GEMPACK Programs

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 5.3.3.

Note that Release 6.0 of GEMPACK provides a way of specifying Stored-input file names (and Log file names) on the command line under DOS - see section 7.3 of GPD-4.

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 a multi-step simulation with a model and you wish to change the shocks. Just edit this file to remove the shock part of the file. Then run the program under option 'asi'. When it comes to the shock part of the input 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.)

5.3.1 Invalid Input When Using Options 'sif' or 'asi'

When you are storing inputs on a file (option 'sif') or adding to an incomplete Stored-input file (option 'asi'), the file you are creating is intended for use as a Stored-input file (under option 'sti') and so must not contain invalid input (as explained above).

Accordingly, when you select either of the options 'sif' or 'asi', option 'bat' is also selected automatically. This means that, if you enter input judged by the program to be invalid, the program will stop with an error (probably "Unexpected input in batch mode"). To continue, proceed as follows.

1. Edit the Stored-input file partly created, taking out the last line (the one containing the invalid input).
2. Rerun the program this time selecting option 'asi' (even if you were previously running under 'sif'). When the program reaches the place just before it stopped, it will transfer control to you and you can give a valid response this time and continue running it.

5.3.2 Differences Between Batch and Interactive Program Dialogues

In this subsection, batch mode means once one of the GEMPACK options 'bat', 'sti', 'sif' or 'asi' has been selected. (It does not include cases such as DOS redirection of input when 'bat' is not 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 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.
- 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/absorption.

5.3.3 Terminal Output and Log Files

When output is going to a Log file and input is being taken from a Stored-input file or a GEMPACK Command file, you may or may not want output also to go to the terminal. GEMPACK allows you to specify this as described below.

1. If, during the Options selection, you select option **log** and later select option **sti** or option **cmf** [52], 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 7.1.2 of GPD-4 for details about the statements

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

5.4 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

[52] 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.

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 sections 5.3-5.9 of GEMPACK document GPD-2. Options for SAGEM and GEMPIE are described in section 7.6 of this document. Some of the options for other GEMPACK programs are documented in chapter 8 below.

The method of selecting options is the same for all programs (and the same whether the option is one of those common to all programs or specific to just one program).

- You select options by typing in their 3-character abbreviation, for example type **LOG** to select the option LOG.
- You can deselect an option by typing a minus sign '-' followed by its 3-character abbreviation, for example type **-LOG** to deselect LOG.
- You can display the current options selected by typing "/"
- Once you have finished option selection you continue to run the program proper by entering a carriage-return. In fact this is usually all you will do since the default options are the most commonly used.

5.5 Increasing Parameters

This section only applies if you are using a Source-code version of GEMPACK in conjunction with a Fortran 77 compiler. With Release 6.0 of GEMPACK, it no longer applies if you are using a Source-code version of GEMPACK with a Fortran 90 compiler, or if you are using an Executable-image version of GEMPACK on a PC. It does not apply if you are using the Demonstration Version of GEMPACK.

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

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

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

This message indicates that some aspect of your model has become too large for the program you were running (as this program is currently configured). Provided you have a source-code licence for GEMPACK[53] and there is enough memory on your computer, it is easy to reconfigure the program. All you have to do is edit the source code of the main program in question, changing the relevant value, and then recompile and relink the program.

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

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

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

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

Note that the programs TABLO and GEMSIM are different from the others in this respect. The parameters are set in the TABLO and GEMSIM INCLUDE files (rather than in the source of TABLO or GEMSIM themselves); for details, see section 2.4 of GPD-2 for TABLO and section 5.13.1 of GPD-2 for GEMSIM. Recompiling and relinking TABLO or GEMSIM is more complicated than for the other programs; again consult your machine-specific documentation and/or your GEMPACK Manager.

The procedure for increasing parameters in a TABLO-generated program is described in section 5.13.2 of GPD-2.

5.6 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.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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 section 5.12 of GPD-2.

5.7 Eliminating Syntax Errors in GEMPACK Command Files

When you run GEMSIM, a TABLO-generated program, or SAGEM, and take input from a GEMPACK Command file, the whole Command file is checked to see that the syntax of all statements (or commands) in it is as expected (that is, conforms to the syntax laid down in chapter 19 of GPD-4). Any syntax errors are pointed out; the message always includes '**%%**' at the start. For example, if a GEMSIM Command file contains the statement "somution file = sjlb ;" (a typing slip), you will see the message

```
%% Unknown keyword 'somution'
```

The program only begins to process (that is, act on) the commands in the Command file if no syntax errors are found.

While this syntax checking is going on, the whole Command file is echoed to the screen, together with any syntax error messages. If you have a large Command file, syntax errors may get lost as the Command file flashes past on the screen. However GEMPACK programs using a Command file always create at least a temporary LOG file (even if you didn't ask for one), and they tell you the name of the LOG file at the end of their run. So, to identify where errors occurred during processing of your Command file, you can search this Log file for '%%' to identify all syntax errors, which you can then correct by editing the Command file. [If you run simulations under WinGEM, it makes it even easier to identify where errors occur.]

5.8 GEMPACK File Types, Names and Suffixes

GEMPACK programs use a number of different file types to communicate between themselves and they produce a number of different types of files containing results for you.

5.8.1 Files with System-determined Suffixes

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 5.8.1: File Type with System-Determined Suffixes

File type	Suffix[54]
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

Whenever a program asks you for the name of any of these files with system-determined suffixes, you should never include the suffix in your input (since the program will add the suffix automatically). For example, in a Command file put

```
Solution file = sjlb ;          ! Correct
```

rather than

```
Solution file = sjlb.sl4 ;      ! Incorrect
```

5.8.2 Suffixes of Other Files

Even though GEMPACK does not force you to use specific suffixes for other files, there are several different types of files where it has become customary amongst experienced GEMPACK users to use certain suffixes. Examples are in the table below.

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

Table 5.8.2: Commonly Used Suffixes for Other Files

File type	Usual Suffix(es)[55]
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 chapter 7 of GPD-4). When you open a Header Array file with ViewHAR, by default it only shows you files with suffixes .DAT, .HAR or .UPD.

Many of the GEMPACK programs suggest suffixes for output files they create. For example, SEEHAR often suggests suffix .SEE for its output files. We suggest that you go along with these suggestions unless you have good reasons to do otherwise.

5.8.3 Files - Binary or Text?

There are two basic file types on all computers - text files (which are sometimes called ASCII files) and binary files. The former can be printed or edited while the latter cannot be. GEMPACK uses several files of each type, as discussed below.

Below is a list of common GEMPACK files (some of which you will not meet until later in this document) 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 cam 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 below), 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 20 of GPD-4.

File Type	Binary or Text
TABLO Input file	text
TABLO Information file	text
Solution file	binary
GEMPIE Print file	text
Equations file	binary
Extrapolation Accuracy file	text
TABLO-generated program	text
GEMSIM Auxiliary Statement,Table files	binary
TABLO-generated Auxiliary files	binary
Environment file	binary
LU file	binary
Base data files	can be binary or text
Updated data files	can be binary or text
GEMPACK Command files	text
Stored-input files	text

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

5.8.4 Why So Many Files?

When GEMSIM or a TABLO-generated program runs, it produces various files.

- The **updated data files** provide information about the simulation and can be used as the starting point for other simulations.
- 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. (If not, you may need to re-run it with more steps.)

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 6.1 below.

More information about the purpose of the other GEMPACK files is given later in this document.

CHAPTER 6

6. Choosing the Closure, Shocks, and Other Sets of Variables

In this chapter we tell you in more detail about choosing the closure for a simulation (section 6.1), choosing other sets of variables such as the variables to shock or the endogenous variables to print (section 6.2) and specifying the shocks for a simulation (section 6.3). All of this chapter applies to multi-step simulations run via GEMSIM or a TABLO-generated program or to Johansen simulations run via SAGEM. Section 6.2 also applies to other programs such as GEMPIE where you are choosing a set of variables from a larger set.

6.1 Specifying the Closure

In the economic models solved via GEMPACK, the number of variables[56] exceeds the number of equations (as we have seen in section 2.11.1 above). In order to solve the equations, some of the variables must be predetermined - the **exogenous** variables - by giving them actual numerical values. This is done by specifying the shocks. From these shocks, the equations of the model can be solved to determine the remaining variables - the **endogenous** variables.

The **closure** of the model, or alternatively the economic **environment** in which you are carrying out a simulation, is the partition (split) of the set of variables into exogenous and endogenous.

For the closure to be valid, the number of endogenous variables must be equal to the number of equations. In addition, the exogenous variables must be an independent set with no equations between them. For example, consider a set of (scalar) variables x,y,z with an equation between them

$$x + y + z = 0.$$

Given x and y as exogenous variables, z cannot also be chosen as exogenous since it is already determined by the equation

$$z = - x - y.$$

The actual choice of exogenous variables depends on what the simulation represents. In general terms variables determined by factors outside the system are often made exogenous, and the remaining variables which are determined by the system being modelled are the endogenous variables.

We recommend that you use GEMPACK Command files for carrying out simulations with TABLO-generated programs or SAGEM.[57] In section 6.1 below we discuss the different ways of specifying the closure on a Command file.

Note that, if you have levels VARIABLEs in your TABLO Input file, you must use the names of the associated linear variables (these have "p_" or "c_" added at the start) when giving names to specify the closure (or when you choose other sets of variables).

---

[56] As explained in section 2.11.1 above, here the number of variables is obtained by adding up the number of components in each of the vector variables. (The VARIABLEs in the TABLO Input file are these vector variables; each has a number of components determined by the sizes of the sets over which their arguments range.)

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

6.1.1 Miniature ORANI Model

Throughout this chapter, examples used are taken from a version of the Miniature ORANI model documented sections 3-9 of Dixon *et al* (1982). For this chapter, the only things you need to know about this model are the names of the sets in this model and the names of the variables in this model, and the number of components each has, as shown below.

SETS

Name	Size	Description	Element names
COM	2	commodities	c1,c2
IND	2	industries	i1,i2
SOURCE	2	source of commodities	domestic,imported
FAC	2	primary factors	labor,capital

VARIABLES

Name	Number of components	Arguments (if any)	Description
XHOUS	4	(COM,SOURCE)	household consumption
CHOUS	1	-	nominal total household consumption
PCOM	4	(COM,SOURCE)	commodity prices
PDOT	2	(COM)	unit cost of Armington aggregate
XDOT	2	(COM)	Armington aggregate over source
U	1	-	consumption mix ratio (household)
PEXP	2	(COM)	export prices (foreign dollars)
PIMP	2	(COM)	import prices (foreign dollars)
XEXP	2	(COM)	export demands
FEXP	2	(COM)	export demand shifters
Z	2	(IND)	industry activity
YCOMIND	4	(COM,IND)	industry output
XINTCOM	8	(COM,SOURCE,IND)	intermediate commod ...
XINTFAC	4	(FAC,IND)	intermediate factor inputs
PLAB	1	-	wage rate
PCAP	2	(IND)	price of capital
PFAC	2	(FAC,IND)	price of primary factors
XINTCOM_CD	4	(COM,IND)	Cobb-D combination of inputs com
PCOM_CD	4	(COM,IND)	price of Cobb-D combination of com
XINTFAC_CD	2	(IND)	Cobb-D combination of factors
PFAC_CD	2	(IND)	price Cobb-D combination factors
V	2	(COM)	power of export subsidy
T	2	(COM)	power of import duty
XLAB	1	-	total demand for labor
XCAP	2	(IND)	industry demand for capital
M	1	-	total imports (foreign or over...
E	1	-	total exports (foreign or over...
CPI	1	-	consumer price index
FWAGE	1	-	wage rate shifter
CR	1	-	real household consumption
XIMP	2	(COM)	import quantities
INTCOM	8	(COM,SOURCE,IND)	intermediate input of com...
INTFAC	4	(FAC,IND)	intermediate input of factor...
HOUSE	4	(COM,SOURCE)	household consumption of com...
EXPCOM_DOMV	2	(COM)	exports of (domestic) commodity...
IMPCOM_DOMV	2	(COM)	imports of commodity (Aust $)
COMPROD	4	(COM,IND)	production of commodity i by ind j
PHI	1	-	exchange rate
B_A	1	-	Aust dollar change in trade bal..
B_F	1	-	foreign dollar change in trade...
EXPSUB	2	(COM)	export subsidies on exports of com
DUTY	2	(COM)	duty levied on imports of com i

If you want to know more about the model, note that the TABLO Input file MO.TAB for this model is included in the model files which accompany GEMPACK - see section 1 of GPD-8. To run Miniature ORANI, run TABLO with the TABLO Input file MO.TAB, in the usual way as described in chapter 2. You can then run the program GEMSIM using the Command file MOTAR.CMF (which is also included in the model files examples) to carry out a tariff simulation with this version of the model.

6.1.2 Specifying the Closure via a Command File

The three main ways of specifying the closure using a Command file are

1. starting from scratch by giving the closure as a list of variable names and components,
2. starting from an existing closure saved on a file and using this closure unchanged, or
3. starting from an existing closure saved on a file and modifying it.

Specifying the closure from scratch
The simplest way of specifying a closure is to give a list of exogenous variables and then specify that the rest of the variables are endogenous. The Command file statements for this are

```
exogenous <list> ;
rest endogenous ;
```

where the <list> is a list of variable names, optionally followed by set/element arguments or component numbers separated by spaces. (Note that these are the names of the associated linearized variables at this stage, since TABLO has already linearized any levels equations into expressions containing percentage changes or ordinary changes of levels variables.)

Individual component numbers can follow the variable names; for example "p_XINTCOM 2-4 6 8" would mean components 2,3,4,6,8 of p_XINTCOM, the percentage change in the variable XINTCOM. Alternatively, components can be indicated using set and/or element names[58] where available; for example, p_XINTFAC("labor",IND) would specify all components of p_XINTFAC with first argument "labor" and second argument in the set IND (that is, all labor inputs to all industries - these are components numbered 1 and 3 of p_XINTFAC).
If a variable name has no set/element arguments and no component numbers following it, this means that all components are being set exogenous.

For example, the following statements specify the usual closure:

```
exogenous p_PIMP p_FEXP p_XEXP("c2") p_PHI
      p_V 1 p_FWAGE p_CR p_T p_XCAP ;
rest endogenous ;
```

Conversely you can list the endogenous variables and make the rest exogenous as in the following statements:

[58] Subset names can also be used. In the example that follows, if there were a subset IND2 of IND declared in the TABLO Input file, then you could write p_XINTFAC("labor",IND2) to indicate all components of INTFAC with first argument "labor" and second argument in IND2.

```
endogenous p_XHOUS p_CHOUS p_PCOM p_PDOT p_XDOT p_U ;
endogenous p_PEXP p_XEXP("c1") ;
endogenous p_Z p_YCOMIND p_XINTCOM p_XINTFAC ;
endogenous p_PLAB p_PCAP p_PFAC p_XINTCOM_CD p_PCOM_CD ;
endogenous p_XINTFAC_CD  p_PFAC_CD ;
endogenous p_V 2 p_XLAB p_M p_E p_CPI p_XIMP ;
endogenous p_INTCOM p_INTFAC p_HOUSE p_EXPCOM_DOMV ;
endogenous p_IMPCOM_DOMV p_COMPROD ;
endogenous c_B_A c_B_F c_EXPSUB c_DUTY ;
rest exogenous ;
```

This would give exactly the same closure as in the first method.

Note that you can continue one statement over several lines (as in the "exogenous ..." statement in the first example above). You can also have several "exogenous ..." or "endogenous ..." statements (as in the second example above).

If you want to **save** this closure on an Environment file so that you can use it later in another simulation, the command

```
save Environment file mo ;
```

can be included. This saves an Environment file MO.EN4 containing the closure specified.

Using a Saved Closure

If the closure used in a simulation has been saved on an Environment file, this file can then be used to specify the closure for a new simulation by using the command in the Command file

```
use Environment file <filename> ;
```

For example, using the Environment file MO.EN4 from the example above, we can reuse exactly the same closure by the command

```
use Environment file mo ;
```

The closure is also written on the Solution file. This file can be used to give the closure instead of the Environment file.

For example,

```
take closure from Solution file <filename> ;
```

is one way of specifying the closure.[59]

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

take closure from LU file <file_name> ;

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

use LU file <file_name> ;

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

Modifying a Saved Closure

If in a simulation you want to have a closure which is very similar to another closure which you have saved, you can start with the old closure and modify it as in the following example from Miniature ORANI.

```
modify closure from Environment file mo ;
swap p_PHI = p_CPI ;
swap p_XEXP("c2") = p_V("c2") ;
```

This starts with the closure on file MO.EN4. In this closure as given in the first example above, p_PHI is exogenous and p_CPI is endogenous. The first swap statement makes p_CPI exogenous and p_PHI endogenous. Similarly the second swap statements makes the previously exogenous component p_XEXP("c2") to be endogenous and the previously endogenous component p_V("c2") to be exogenous.

Another example is the swapping of several components of one variable with several components of another variable as in the following:

```
modify closure from Environment file mo ;
swap p_PIMP 1 2 = p_PCOM 1 4 ;
```

Note that, in a "swap" statement, if all components on one side are exogenous, those on the other side must all be endogenous (and vice versa). The total number of components on each side must be the same. These components are interchanged by "swapping" the sets to which they belong.
You can also use "exogenous ... ;" and "endogenous ... ;" statements to modify a closure. For example, the following statements would specify the same modified closure as those in the first example above.

```
modify closure from Environment file mo ;
endogenous p_PHI p_XEXP("c2") ;
exogenous p_CPI p_V("c2") ;
```

The full syntax for these commands and other similar commands is given in chapter 19 of GPD-4. Examples of complete Command files are given in sections 2.8.1 and 2.10.4 and in sections 19.2.3 and 19.3.1 of chapter 19 of GPD-4 and in the model files listed in section 1 of GPD-8.

6.1.3 Component Numbers for a Variable

You will need to know the order of the components of a VARIABLE if you want to specify component numbers in the statements giving the closure (as above) or in selecting sets of variables (section 6.2 below) or in specifying shocks (section 6.3 below).

A VARIABLE with no arguments has just one component. A VARIABLE with one argument (for example, p_PEXP(i) in Miniature ORANI) has as many components as the size of the set over which the argument runs; and these components are in the same order as the elements of this set.

For VARIABLEs with 2 or more arguments, the order of the components is determined by the rule that the first argument varies fastest, then the second varies next fastest, and so on.

The examples below (all taken from Miniature ORANI as described in section 6.1.1 above) should make this clear.

Examples

1. p_XEXP has one argument ranging over the set COM.

```
Component 1     p_XEXP("c1")
Component 2     p_XEXP("c2")
```

2. p_YCOMIND has two arguments ranging over COM and IND.

```
Component 1     p_YCOMIND("c1","i1")
Component 2     p_YCOMIND("c2","i1")
Component 3     p_YCOMIND("c1","i2")
Component 4     p_YCOMIND("c2","i2")
```

3. p_XINTCOM has three arguments ranging over COM, SOURCE and IND respectively. The first 4 components correspond to the first industry "i1" and the last 4 to the second industry "i2" since the last argument varies slowest.

```
Component 1     p_XINTCOM("c1","domestic","i1")
Component 2     p_XINTCOM("c2","domestic","i1")
Component 3     p_XINTCOM("c1","imported","i1")
Component 4     p_XINTCOM("c2","imported","i1")
Component 5     p_XINTCOM("c1","domestic","i2")
Component 6     p_XINTCOM("c2","domestic","i2")
Component 7     p_XINTCOM("c1","imported","i2")
Component 8     p_XINTCOM("c2","imported","i2")
```

6.2 Choosing Other Sets of Variables

There are many situations in which you must choose a set of variables. For example,

1) when carrying out a multi-step simulation, you must choose the endogenous variables to be retained on the Solution file (the so-called "cumulatively-retained endogenous variables"), or
2) when printing totals results via GEMPIE, you must choose the endogenous variables to be in the printout.

In such situations, there is always a "big" set and you are choosing a subset of this big set. In example (1) above, the big set is the set of all endogenous variables (including any backsolved for if you condensed the model) while in (2) above it is the set of cumulatively-retained endogenous variables.

In GEMSIM, TABLO-generated programs and SAGEM, you can choose sets of variables using a Command file. This method is illustrated in section 6.2.1.

In choosing sets of variables in other programs, in particular GEMPIE, Command files are not available so you may need to choose sets of variables interactively (or by using a Stored-input file). Some details about this are given in section 6.2.2 below. Complete details of interactive choice of sets of variables are given in chapter 21 of GPD-4.

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

6.2.1 Choosing Sets of Variables via a Command File

There are two choices of sets of variables for **GEMSIM or TABLO-generated programs**, namely:

1) Cumulatively-retained endogenous variables which are chosen from the set of all endogenous variables (including any backsolved for if you condensed the model),
2) The set of variables which are on the Extrapolation Accuracy file (that is, the XAC-retained variables), chosen from the set of all endogenous variables (including any backsolved for if you condensed the model).

For **SAGEM**, there are various choices to make, including:

1) Individually-retained exogenous variables from the set of all shocked exogenous variables,
2) Individually-retained endogenous variables from the set of all endogenous variables,
3) Cumulatively-retained endogenous variables from the set of all endogenous variables.

The meanings of individually-retained and cumulatively-retained are explained in detail in chapter 7. Here we will just assume that in each choice, we have a "big" set and are choosing a subset of this set. The Command file statements are similar for all these choices so we will give them all together and illustrate by examples the form of the <list> used in specifying them:

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

In each case, the <list> must consist of variables and components in the "big" set. The things in the <list> can be

1) a variable name followed by set and/or element arguments, for example

   p_XINTFAC("labor",IND)[60]

   which means all components of p_XINTFAC with first argument "labor" and second argument in the set IND (that is, all labor inputs to all industries - these are components numbered 1 and 3 of xINTFAC); this means that the components indicated are to be included in the subset. (They must all be in the "big" set.)

2) a variable name followed by certain components of it (these must all be in the big set), for example,

   p_XINTCOM 2-4 6

   which means that just these components are to be included in the subset.

3) a variable name not followed by set/element arguments or component numbers, which means that all components of this variable which are in the big set are to be included in the subset.

4) abbreviations such as

%all	meaning all variables in the big set, or
%macro or **%scalar**	meaning all macro or scalar variables in the big set are to be included in the subset.

5) abbreviations such as

(FAC)	meaning that all variables with one argument ranging over the set FAC which are in the big set are to be included in the subset, or
(COM,SOURCE)	meaning the same for variables with two arguments, the first ranging over COM and the second ranging over SOURCE.

[60] There must not be a space between the end of the variable name and the '('. For example, "p_PEXP(COM)" and "p_PEXP (COM)" mean quite different things - see (5) below for the meaning of "(COM)" in the second case.

Example using SAGEM

Suppose shocks are to be applied to the following exogenous variables. (We are assuming the closure is as saved on the Environment file MO.EN4 - see the section headed "Specifying the closure from scratch" in section 6.1.2 above.)

p_PHI, component 2 of p_T, p_FWAGE and p_CR.

```
!
! Command file for running SAGEM for the Miniature ORANI model
!  from TABLO Input file MO.TAB
!
use equations file  mo ;
solution file = mosagem ;
!
! Closure
!
use environment file mo ;
!
! Shocks
!
shock p_T 2 =  1 ;
! or 'shock p_T("c2") = 1 ;'
shock p_PHI = 1 ;
shock p_FWAGE = -2.38 ;
shock p_CR = 2.76 ;
!
! Choosing sets of variables
!
individually-retained exogenous %all ;
individually-retained endogenous p_Z p_YCOMIND(COM,"i2")
                                     p_XINTFAC 1-3 %macro ;
cumulatively-retained endogenous p_Z p_YCOMIND (COM) (FAC,IND) ;
!
!
verbal description =
MO Standard closure.
Shocks  p_phi p_t 2 p_fWAGE p_cR ;
!
!  End of Command file
!
```

The three commands above choosing sets of variables determine what is on the Solution file and hence what you can print out later using GEMPIE. The first two commands - see (1) and (2) below - refer to individual solutions on the Solution file while command (3) refers to the totals solution.

(1) The command

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

means that the Solution file will show results for all shocks. There will be a separate column for each of the shocks. (The big set here is the set of all shocked variables.)

(2) The command

```
individually-retained endogenous p_Z  p_YCOMIND(COM,"i2")
      p_XINTFAC 1-3 %macro ;
```

means that the Solution file will contain results of these shocks on

variable p_Z (all components)

components (c,"i2") for all 'c' in COM of p_YCOMIND

components 1,2,3 for variable p_XINTFAC

all endogenous macro variables (that is, variables with just one component) -
p_CHOUS, p_U, p_PLAB, p_XLAB, p_M, p_E, p_CPI, c_B_A, c_B_F

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

(3) The command

`cumulatively-retained endogenous p_Z  p_YCOMIND  (COM)  (FAC,IND) ;`

means that the Solution will contain the cumulative effect of all the shocks on the following.

variables p_Z and the specified components of p_YCOMIND.

(COM) chooses all (endogenous) components of all variables

in the model having one argument ranging over the set COM

- p_PDOT, p_XDOT, p_PEXP, p_XEXP, p_V, p_XIMP,
p_EXPCOM_DOMV, p_IMPCOM_DOMV, c_EXPSUB and c_DUTY.

(FAC,IND) chooses all variables in the model having two

arguments ranging over the sets FAC and IND (in that order)

- p_XINTFAC, p_PFAC and p_INTFAC.

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

6.2.2 Choosing Sets of Variables Interactively

Where Command files are available, we strongly recommend that you always use them since making choices interactively (or even on a Stored-input file) is less reliable and less transparent. However, sometimes you will need to make choices using the methods described here (for example, if you wish to choose the variables whose results appear on a GEMPIE Print file).

You will recognise the choice situations we are describing in this section because you will be presented with a menu looking something like that shown below.[61] (In the menu below, the words "shocked exogenous" and "individually-retained exogenous" will be replaced, more generally, by words describing the big set and the subset being chosen.)

```
     CHOICE OF WHICH shocked exogenous VARIABLES YOU WANT
                  TO BE individually-retained exogenous.
  Make ONE of the following choices:
   L     LISTS of variables, all or some of whose
          components are to be individually-retained exogenous.
   a     ALL components of ALL shocked exogenous
          variables to be individually-retained exogenous.
   n     NO shocked exogenous
            variables to be individually-retained exogenous.
   m     All shocked exogenous MACRO
            variables to be individually-retained exogenous.
   1     All components of ONE shocked exogenous
          variables to be individually-retained exogenous.
   f     All components of a FEW shocked exogenous
          variables to be individually-retained exogenous.
   s     SOME components of SOME shocked exogenous
          variables to be individually-retained exogenous.
   w     WHICH are the shocked exogenous variables.
  Enter your choice now. ('L' is the default.)
```

Example of the Menu Presented for Choosing a Subset

Sometimes your choice will be simple and easily expressed. For example, you may wish to choose all variables or none, or all "macro" variables (that is, variables with just one component), or just a few variables. Procedures for doing these are described in section 6.2.3 below.

On the other hand, you may need to describe a fairly complicated set - perhaps many variables and only some of the components of some of them. Procedures for making such choices are described in section 6.2.4 below.

The rest of this section is a description of the most important ways of choosing a set of variables interactively. A fuller treatment can be found in chapter 21 of GPD-4.

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

6.2.3 Simple Choices

The simple choices, and their meanings are shown below.

Select

```
a (ALL)     if the subset is identical to the big set.
n (NO)      if the subset contains no variables.
m (MACROS)  if the subset consists of all macro variables
            which are in the big set. A macro variables has
            exactly one component.
```

If you select one of these menu items, this completes the choice of subset. (If you want to choose all macro variables **and** some other variables, choose 'L' as in section 6.2.4.)

Select

```
1 (ONE)     to choose all components (which are in the big set)
           of just one variable. You will then be asked for
            the name of this one variable.  .
```

Note that only the components of a variable which are in the big set are available for the subset. If, for example, a variable 'x1' has 10 components of which only numbers 1-5 and 10 are in the big set and you select '1' and then nominate 'x1', only these 6 components will be in the subset.

Select

```
f (FEW)   to choose all components (which are in the big set)
          of a few variables. You will then be asked how many
          variables and then for the names of these variables
          (one per line).
w (WHICH) for a description of which components of which
          variables are in the big set. Then you can choose
          again from the menu above.
```

6.2.4 Choosing Using the Lists Option

The option **L** ("Lists") is the best way of choosing complicated sets of variables interactively. (The option **s** is another way - see section 21.3 of GPD-4 for details.) Once you have chosen **L** from the main menu, you can use several commands, shown in the menu below to specify the chosen set.

Note that, as you issue more commands, you keep adding to the subset until you choose **f** to finish.

```
COMMAND            MEANING
a  <list>  Choose ALL  components of the listed variables.
s  <list>  Choose SOME components of the listed variables.
m          Choose all available MACRO variables.
w          Which are available components of variables to choose from?
c          Which components of which variables have I already chosen?
f          Finish.
q          Quit.(Give up this attempt to choose a set of variables.)
?          Help.
?  <list>  Detailed help for the specified commands.
```

Subcommands Available When Choosing via the Lists Option

Example of choosing interactively after choosing L

In the Miniature ORANI model, suppose we wish to choose

1. all components of p_Z,
2. just components p_YCOMIND(c,"i2") for all 'c' in COM,
3. components 1,2,3 of variable p_XINTFAC, components 1 and 4 of p_XINTCOM,
4. all macro variables, and
5. all variables with one argument where this argument belongs to the set COM. Also all variables with two arguments, the first belonging to set IND and the second to FAC.

The responses for these choices are as follows:

1. To choose all components of p_Z (that are in the big set), enter

   a p_Z

2. The response below will choose the desired components of p_YCOMIND.

   a p_YCOMIND(COM,"i2")

3. After this, a response of

   s p_XINTFAC p_XINTCOM

   followed by, (responding to prompts about p_XINTFAC and p_XINTCOM)

   3 (for 'p_XINTFAC')
   1-3
   2 (for 'p_XINTCOM')
   1 4

   will say that just components numbered 1-3 of 'p_XINTFAC' and components 1 and 4 of 'p_XINTCOM' are to be chosen.

4. To choose macro variables, enter
   m

5. You can choose whole classes of variables by indicating their arguments and the sets over which they range. For example

   a (COM) (IND,FAC)

   chooses all (available) components of
   (i) all variables in the model having one argument ranging over the set COM (this is what **(COM)** means), and
   (ii) all variables in the model having two arguments ranging respectively over the sets IND and FAC (this is what **(IND,FAC)** means). (Of course, in such cases, the sets and arguments are as contained on the TABLO Input file for the model.) The set name or names must be enclosed in brackets **(** and **)** in this type of command.

6. Then give the command which completes the choice of the subset.
   f (to finish)

6.3 Specifying the Shocks

If you are using a Command file, you specify the variables to be shocked and the numerical values of the shocks at the same time, as explained in section 6.3.1 below. If you are running interactively, you could specify the variables to be shocked following the procedure described in section 6.2.2 above and then specify the numerical values of the shocks separately, as explained in section 6.3.2 below. (As before, always use names of linear VARIABLEs when specifying variables to be shocked.)

Note that the shocks to a particular variable are said to be **uniform** if all shocked components of this variable are given the same numerical shock.

6.3.1 Specifying the Shocks via a Command File

To specify the shocks to exogenous variables using a Command file, you must have one shock command for each variable to be shocked. All or some components of this variable can be shocked.

If only a few shocks are involved, a simple list of components and corresponding shocks can be given. One example of this type of command is given in the Command file in section 6.2.1.

```
!
! Shocks
!
shock p_T 2 =  1 ;  ! or 'shock p_T("c2") = 1 ;'
shock p_phi = 1 ;
shock p_fWAGE = -2.38 ;
shock p_cR = 2.76 ;
```

You can specify shocks to several numbered components of a variable. For example if p_XINTFAC were exogenous and we wanted to shock components 1, 2 and 4, the command

```
shock p_XINTFAC 1 2 4 = 0.2 0.4 -3.9 ;
```

means that component 1 of XINTFAC is shocked by 0.2, component 2 by 0.4 and component 4 by -3.9.

Alternatively the components to shock can be indicated using sets and/or elements as arguments. For example,

```
shock p_XINTFAC("capital",IND) = 4 6 ;
```

means shock p_XINTFAC("capital","i1") by 4 per cent and shock p_XINTFAC("capital","i2") by 6 per cent.

For a uniform shock applied to just components 1, 2, 4 of p_XINTFAC, the command is

```
shock p_XINTFAC 1 2 4 = uniform 3.0 ;
```

or to shock just components relating to capital inputs uniformly,

```
shock p_XINTFAC("capital",IND) = uniform 3.0 ;
```

or, if all components of XINTFAC are shocked uniformly,

```
shock p_XINTFAC = uniform 3.0 ;
```

When you are giving a large number of shocks to a variable, you can use a **shock file**.
For example,

```
shock p_XINTFAC = file FACTOR.SHK ;
```

tells the program to read the shocks from the file FACTOR.SHK into the components of variable p_XINTFAC. The method of preparing shock files is discussed in section 6.3.3.

There are situations in which you have computed on a file (perhaps a file written by GEMSIM or a TABLO-generated program) possible shocks to all components of a variable, but only want to apply some of these shocks in the current simulation. In such a case, you do not have to edit the file to remove the values corresponding to the non-shocked components; instead you can direct the program to select the values on the file corresponding to the components of interest. For example, the statement

```
shock p_XINTFAC("capital",IND) = select from file FACTOR.SHK ;
```

will shock only the two components ("capital","i1") and ("capital","i2") of p_XINTFAC and read the relevant values from the file FACTOR.SHK. In this case, the file FACTOR.SHK must have 4 values on it, and the second and fourth are read (since the components mentioned are components number 2 and 4 of variable p_XINTFAC - see section 6.1.3 above). The words
"**select from file**" are what distinguishes this from other cases.

Similarly you can use "select from" followed by a list of values, as in, for example,

```
shock p_XINTFAC("capital",IND) = select from 3 4 5 6 ;
```

Here components ("capital","i1") and ("capital","i2") of XINTFAC will be shocked by 4 and 6 per cent respectively.

6.3.2 Specifying the Shocks Interactively

Since Command files are always available when you must specify shocks, we strongly recommend that you always use them since specifying shocks interactively (or even on a Stored-input file) is less reliable and less transparent

The steps involved in specifying shocks interactively are roughly as follows. Prompts are given at all steps. You are first asked

Are all shocks are equal to 1? [y/n]

If you respond 'y' (yes), this completes the specification of the shocks.

If you respond 'n', you are presented with a list of the relevant variables (namely, those with at least one component shocked) in turn, one at a time. They are presented in the order in which the variables are declared in the TABLO Input file for the model.

For a variable with only one component shocked, you are asked to enter the numerical shock.

For a variable with more than one component shocked, the choice is a little more complicated.

You are first asked if you want to give **uniform** shocks (that is, the same shock to each shocked component). If you respond 'y' (yes), you are next asked to enter the value of this uniform shock.

If you respond 'n' (not uniform shocks), you are asked whether you want to read the shocks from a file (a text file) or the terminal. In the case of a file, you are then asked for the name of the file, after which the program reads the values from that file; see section 6.3.3 below for advice about preparing these files. In the case of the terminal, you are then asked to enter the appropriate number of shock values.

This is repeated for each variable with at least one component shocked.

6.3.3 Shock Files

If you specify a file from which to read the shocks for one variable, this file must be a GEMPACK text file. It must begin with the "how much data" information which, on shock files, can usually take the simple form

<number> ;

where <number> is the number of numerical values of shocks on the file and the ending semicolon ';' is required. In general, <number> is the number of shocked components of the variable in question. However when you are selecting shocks from a file using the "select from file.." statement as described in section 6.3.1, <number> is the total number of components for the variable.
Then, on the subsequent line (or lines, as you please) must come <number> real numbers, being the values of the shocks.[62]

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

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

These numbers can be separated by spaces or commas.

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

<integer>*<value>

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

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

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

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

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

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

In this example, the file might be as below.

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

Shock files must conform to the syntax of GEMPACK text data files, as documented in chapter 20 of GPD-4.

---

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

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

CHAPTER 7

7. More About Simulations and Solution Files

In this chapter we go into more detail about the contents of Solution files, both those produced by SAGEM and those produced after a multi-step simulation. In doing so for SAGEM, we introduce the idea of a subtotal solution. We also describe in a little more detail how the linearized equations are solved, and give information on and advice about choosing the Harwell parameter required by the sparse matrix routines used by GEMPACK. In section 7.6 we describe the Options available for SAGEM and GEMPIE.

7.1 Several Simultaneous Johansen Simulations via SAGEM

As explained in section 2.10 above, it is possible to calculate the results of several Johansen simulations in one run of SAGEM. In this section we explain this in a little more detail.

As we have seen in section 2.11.2 above, once the exogenous/endogenous split has been chosen, the system of linearized equations of a model with n equations and m variables

$$\mathbf{Cz = 0} \qquad (1)$$

becomes

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

where z_1 and z_2 are respectively the vectors of endogenous and exogenous variables in the system,

A is n x n and D is n x (m - n).

The columns of A and D are those columns of C corresponding respectively to endogenous and exogenous variables. If k of the m - n exogenous variables are given nonzero shocks, the problem of finding the change in each of the n endogenous variables resulting from each of these k shocks reduces to solving, for X, the matrix equation

$$\mathbf{AX = B} \qquad (3)$$

where B is an n x k matrix, the jth column of which is obtained by multiplying the column of D corresponding to the jth shocked variable by the negative of the shock given to this variable.

When the closure is economically sensible, the matrix A is invertible and the equation (3) can be solved.

The n x k matrix X is the "solution" of the simulation: its entry

`x(i,j) (in row i and column j)`

shows the percentage change (or change) in the ith endogenous variable resulting solely from the shock given to the jth shocked variable. Because we are dealing with a system of linear equations in (3), the sum of all entries in any row i of X can be taken as the percentage change in the ith endogenous variable resulting from **all** of the k shocks given. The elasticity of the ith endogenous variable with respect to the jth shocked variable can be obtained by dividing x(i,j) by the shock given to the jth shocked variable. (The first two columns of results in Table 2.10.2 in section 2.10.2 above are the X matrix, of size 27 x 2, for the simulation carried out there.)

The Solution Matrix X and Totals Column T

	Shocked variables				
	1	2	...	k	Totals column
Endogenous variables					
1	[x(1,1)	x(1,2)	...	x(1,k)]	[t(1)]
2	[x(2,1)	x(2,2)	...	x(2,k)]	[t(2)]
:	:	:	...	:	:
:	:	:	...	:	:
n	[x(n,1)	x(n,2)	...	x(n,k)]	[t(n)]
	Solution matrix X				**Totals column T**

When SAGEM is used in this way, the Solution file produced contains selected portions of the solution matrix X. The exogenous variables corresponding to the columns of X retained on the Solution file are called the **individually-retained exogenous** variables. (There may be none of them.) The endogenous variables corresponding to the rows of X retained in the Solution file are called the **individually-retained endogenous** variables. If any of the individual columns of the matrix X are retained on the Solution file (that is, if there are any individually-retained exogenous variables), we say that the Solution file contains **individual column results**.

It is also possible for the Solution file to contain all or some of the rows of the n x 1 vector T whose ith entry (for i between 1 and n) is obtained by adding up all the entries in row i of the matrix X. The endogenous variables corresponding to the rows of T (if any) retained on the Solution file are called the **cumulatively-retained endogenous variables** since the ith entry of T is the cumulative effect on the ith endogenous variable of all the shocks applied. If some of T is retained on the Solution file, we say that the Solution file contains **cumulative** (or **row totals**) **results**. (The third column of results in Table 2.10.2 in section 2.10.2 above is the T vector for the simulation carried out there.)

When running SAGEM, if you choose to retain some of the rows of the totals vector T, you can also store on the Solution file selected **subtotals** which are the cumulative effect of several (but not necessarily all) of the shocks. We call these **subtotal solutions**. Each of these can be thought of as an n x 1 vector whose ith value is obtained by adding all entries in row i of X which are in columns corresponding to the shocks in question. The rows kept on the Solution file for each subtotal are the same rows as those kept from T, namely those corresponding to the cumulatively-retained endogenous variables. More details about obtaining subtotals results (which can also be obtained when running GEMPIE) are given in section 7.3 below.

A Solution file produced by SAGEM may contain both individual column and totals/subtotals results, or just one of these two types, according to the instructions you issue while running SAGEM.

In summary, the Solution file produced by SAGEM contains

- selected rows and columns of the solution matrix X, showing the effects on selected endogenous variables of selected shocks, and/or
- selected row totals (or subtotals) across all (or some) columns of X, showing the total effect, accumulated over all (or some) of the shocks, on selected endogenous variables.

You control the contents of the Solution file (when running SAGEM) by specifying

a) whether the Solution file contains individual results, totals/subtotals results or both types of results,

b) the individually-retained exogenous variables,

c) the individually-retained endogenous variables,

d) the cumulatively-retained endogenous variables,

e) which subtotals (if any) and the effects of which shocks are included in each subtotal solution.

We strongly recommend that you always use a Command file when running SAGEM. [Running SAGEM interactively is error-prone and difficult to document or reproduce.]

7.2 Printing the Results of a Simulation

When you run GEMPIE you can translate to printable form any results stored on the Solution file you access. If the Solution file contains both individual column results and totals/subtotals results, you can only convert part of one of these in any one run of GEMPIE. If you want to print both sorts of results, you must run GEMPIE twice.

Alternatively, if the Solution file contains individual column results, you can use these individual column results to form up subtotals results (the results of all or some of the shocks whose individual effects are stored) and then convert these subtotals results to printable form. More details of this are given in section 7.3 below.

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

7.3 Subtotals

There are two ways of forming subtotals results; the first is when running SAGEM and the second is when running GEMPIE. We elaborate on these below. First we give in section 7.3.1 an example and use it to illustrate these two different ways. Then, in sections 7.3.2 to 7.3.4, we give more general information relevant to subtotals.

7.3.1 Subtotals - An Example

Suppose you shock

- the foreign currency prices [p(i),i=1,...,10] for the imports of each of the 10 commodities by 5 per cent,
- the real wage rate [w] by 2 per cent, and
- household consumption [c] by 1 per cent.

Then X would have 12 columns (10 for the p(i) shocks and one each for the w and c shocks). The combined effect of just the p(i) shocks would be given by adding just the first 10 columns of X - this is one subtotal you may wish to calculate and print. A second subtotal solution of the model would be the combined effects of the w and c shocks.

FIRST METHOD - VIA SAGEM

When you run SAGEM you can ask it to store on the Solution file all of the following:

a) the 12 individual columns,

b) the cumulative total T,

c) the subtotal solution showing the combined effect of the 10 p(i) shocks,

d) the subtotal solution showing the combined effect of the w and c shocks.

Then, when you run GEMPIE, you will be able (in one run)
either to

- print some or all of the individual columns, **or**
- print the cumulative solution column T and 0, 1 or 2 of the subtotal solutions on the Solution file.

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

SECOND METHOD - VIA GEMPIE

Alternatively, you may run SAGEM and just store the 12 individual columns (but no cumulative total or subtotals) on the Solution file. You can still obtain the same subtotal solutions by running GEMPIE and specifying two subtotals to be printed, the first summing over the 10 price shock columns and the second over the w and c shocks.

In the second method, the subtotals are not stored on the Solution file but must be defined when running GEMPIE. This method is suitable when you do not know in advance what subtotals you may require from a particular simulation.

7.3.2 Subtotals and Sets of Shocks in General

For a given simulation, you can have several different subtotals. Each is associated with a subset of the shocks. To specify a subtotal, you must say which of the shocked variables you want the subtotal to reflect. (This specifies which columns of the solution matrix X are to be added up to produce the subtotals column.)

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

7.3.3 Storing Subtotals on Solution Files When Running SAGEM

If you choose to keep any cumulatively-retained endogenous variables, you will be asked if you want to store any subtotals on the Solution file. Note that if you do not keep any cumulatively-retained endogenous variables, you will NOT be able to store any subtotals. If you respond 'y' (yes) you will be asked to describe the first subtotal you have in mind by specifying exactly which shocks to sum over. (You do this by the Lists method, as introduced in section 6.2.4 above and described fully in section 21.2 of GPD-4.) Then you will be asked for a one-line description of this subtotal. By now you have specified just one subtotal. After this (and any subsequent subtotals) you will be asked if you want any more subtotals. You can repeat the above procedure for as many subtotals as you require. Note that the endogenous variables retained in each and every subtotal solution are the same as those kept in the cumulative column, namely the cumulatively-retained endogenous variables.

Any subtotals stored on the Solution file can be printed when running GEMPIE (as described below).

7.3.4 Printing Subtotals or Individual Column Results using GEMPIE

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

1) Print [63] any subtotals solutions (and the cumulative solution) stored on the Solution file.
2) Print any individual column results stored (by SAGEM) on the Solution file.
3) Setup and print new subtotals calculated from the individual column results stored (by SAGEM) on the Solution file.

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

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

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

1) If you choose to print subtotals already stored on the Solution file, you will be asked which ones you wish to print. If you only want some of the subtotals stored to be printed, you indicate which ones by number (for example, 1-3, 5 to indicate 4 of the 6 available subtotals solutions). The subtotals you select will be printed in columns side-by-side and the cumulative solution will be printed as the last column.
2) If you choose to print individual column results, you will be asked to specify the columns (which are a subset of the individually-retained exogenous variables) and the rows (which are a subset of the individually-retained endogenous variables) to print.
3) If you choose to setup and print new subtotals, you will first be asked for the endogenous variables to be printed in each subtotal. (These are a subset of the individually-retained endogenous variables.) Then you specify the subtotals, one at a time, following essentially the same method as that used for creating subtotals when running SAGEM (see section 7.3.3 above). Note that, when running GEMPIE, the subtotals are summed over some or all of the individually-retained exogenous variables (the only ones available on the Solution file) whereas in SAGEM they are summed over some or all of the shocked variables. Thus, in setting up one subtotal when running GEMPIE, you are choosing a subset of the individually-retained exogenous variables.
   Note that subtotals solutions you set up and print here are not stored on the Solution file by GEMPIE (which never alters a Solution file).

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

7.4 Solution Files from Multi-Step Simulations

Although it is possible to carry out several Johansen simulations (all with the same closure) simultaneously in one run of SAGEM, it is not possible to carry out several multi-step simulations in one run of GEMSIM or the appropriate TABLO-generated program, as explained below.

For Johansen simulations, the matrix A in (2) of section 7.1 above depends only on the closure and is not affected by the shocks. Since the main computing cost of a Johansen simulation is that of calculating the LU decomposition of A, it is possible to carry out several different simulations (all with the same closure but different sets of shocks) in one run of SAGEM. Certainly this increases the number of right-hand-sides to solve for, but the solution for each of these is relatively cheap to compute.

In contrast, when you give a set of shocks in say a 2-step simulation (see Figure 2.11.3 in section 2.11.3), the program first applies half of each shock, then updates the data to reflect the effects of these, then (for the second of these steps) recomputes the matrix C (as in equation (1) in section 7.1 above) and solves for the effects of the next half of the shocks in question. For 2 different sets of shocks (that is, for two different multi-step simulations), you would get two different C matrices after the first half of each shock has been applied. (In the notation of Figure 2.11.3, if you have two sets of shocks you will reach two different points C2 after the first step and then move in two different directions from these in the second step.) This means that there are two different A matrices to LU decompose, and it is not feasible to carry on these two quite separate calculations side-by-side.[64]

Of course, this problem would become even less tractable for larger numbers of steps.

For this reason, the Solution file from a multi-step simulation contains just one totals solution (the cumulative effect of all the shocks). No individual column (or other subtotal) results can be present.

In summary, SAGEM can produce a lot of approximate results. But if you want accurate solutions to the (usually nonlinear) levels equations of your model, you can only calculate one solution at a time.

[64] Two different sets of shocks with the same closure give rise to the same A matrices in the first step of a multi-step simulation. When computing an extrapolated solution from, say 1-step and 2-step solutions, GEMSIM or TABLO-generated code takes advantage of this by computing the solution of the first step of the 2-step computation immediately after computing the solution of the first step of the 1-step computation, both using the same LU decomposition.

7.5 How SAGEM and TABLO-generated Programs Solve Linear Equations

Whether you are doing a multi-step simulation via GEMSIM or a TABLO-generated program, or a Johansen simulation via SAGEM, a critical part of the calculation is to solve a system AX=B of linear equations, where A is the n x n matrix in equation (3) in section 7.1 above. In the case of SAGEM, B and X are n x k if you are solving for k individual column results, while, for TABLO-generated code, X and B always consist of a single column (since only one solution is being calculated). In a p-step simulation, p matrix equations like this must be solved, one at each step.

In the systems of linear equations AX=B (see (3) in section 7.1) that arise from general equilibrium models, the matrix A is a sparse matrix in the sense that only a small proportion of its entries are nonzero. (This general property of CGE models is explained in section 8.1 below.) GEMPACK solves this matrix equation using the Harwell Laboratories sparse matrix routine MA28 (see Duff (1977)). This does not explicitly invert the matrix A but calculates the so-called **LU decomposition** (see, for example, section 8.1 of Atkinson (1989) or Stewart (1973)) of A. This is a very efficient procedure which allows very large models to be handled directly by GEMPACK.

One feature of the Harwell sparse matrix routines is that there is a trade-off between maintaining sparsity during the calculation and minimising the effects of rounding errors, which are inevitable when doing a large computation on a computer. The Harwell routines ask you to supply the value of a parameter U, referred to as the **Harwell parameter**, which must be between zero and one. The precise consequences of choosing different values for this parameter are difficult to predict. However some general comments can be made.

- Choosing U close to one minimises the rounding errors, so maximises the accuracy of the solution. However this also increases the time required to produce the solution.
- Choosing U small (say equal to 0.1) decreases the time taken to calculate the solution but may slightly increase the rounding errors (and hence slightly reduce the accuracy of the results).

We are often asked how many figures of accuracy will be lost if U is taken to be 0.1. Unfortunately, it is not possible to answer this precisely; all we can say is that, as far as we can tell, the loss of accuracy is not great (certainly taking U equal to 0.1 should not lose one figure of accuracy in all results but it may do so in some of the results).

Fortunately, it is not of great importance since, in Release 5 of GEMPACK, both SAGEM, GEMSIM and TABLO-generated code use a well-known **iterative refinement** technique called residual correction of solutions (see, for example, section 8.5 of Atkinson (1989)) when solving AX=B. Our experience is that this compensates very effectively for any loss of accuracy that may have resulted in its absence from using small values of U.[65]

Accordingly we are confident in recommending that

> **U be taken as 0.1 in all but those very small models for which computation time is not an issue.**

(This is why U=0.1 is the default for Command files - see chapter 19 of GPD-4.)

[65] In one spectacular instance, we tested a very large model whose homogeneity seemed uncertain judging by results produced with U=0.1 without using residual correction. When U was increased to 0.5, the results were as desired but the running time increased from hours to days. When we tested residual correction with this model, we found that, after residual correction, the U=0.1 solution was at least as accurate as the U=0.5 result; further, the residual correction added only a few seconds to the computation time.

7.6 Special Options for SAGEM and GEMPIE

Several options are available for SAGEM and GEMPIE apart from the basic options BAT, BPR, LOG, STI, SIF and ASI (which have been described in chapter 5). For SAGEM, these are CPU, CMF, NSM, NIR, NWE and KZC while for GEMPIE, there are options CPW, RPO and CPL, OSR, OSS, SNA, NBV, FBS. These extra options will be described below. However you will find that **the default options** (that is the program settings you get if you do not choose any options) are suitable in most standard cases.

Note that, as for any other GEMPACK program, **online help** is available. For example, to find information about the SAGEM option **NIR**, just respond '**?NIR**' when prompted for your choice of options. Responding '**??**' will give online help for all options.

7.6.1 SAGEM Options

SAGEM can be run using a GEMPACK **Command file**. This is a text file which consists of a sequence of brief commands to give details such as file names, closure and shocks, and a verbal description of the simulation. Select option

CMF Input from a GEMPACK Command file

and give the name of the Command file. Further details are given in section 2.10.2 and in section 19.2 of GPD-4 about preparing Command files for SAGEM and their syntax.

Usually SAGEM carries out iterative refinement of the solutions to improve their accuracy. However you can turn iterative refinement off using option

NIR Don't do solution iterative refinement

If you only wish to set up the closure for a simulation and then save it on an Environment file, select

NSM No simulation (closure only)

The Environment file created can then be used in later simulations to specify the closure.

On most machines, it is possible to obtain details of the CPU time (actual machine or Central Processing Unit time). Select option

CPU Report CPU times

to obtain times for the five stages of a simulation:

1) User specification of the simulation,
2) Creation of the right-hand side matrix, that is, the matrix B in equation (3) of section 7.1,
3) Creation of the left-hand side matrix, that is, the matrix A in equation (3) in section 7.1,
4) LU decomposition of the left-hand side matrix, and
5) Calculation and writing of solutions.

However, on some machines, CPU times are always reported as zero, which means that CPU reporting is not available within GEMPACK on this machine.

Normally warnings are given to indicate how well the equations have been solved. Warnings are given if the left hand side of the equation differs from the right hand side, or if the Equations Matrix is ill-conditioned. Option

NWE Don't warn how well equations solved

fill suppresses these warnings. (See section 5.2.6 of GPD-2 for more information about these warnings.)

The final option for SAGEM is

KZC Keep zero coefficients

Zero coefficients are entries in the Equations Matrix which are calculated to be zero from the current data base but may be non-zero for some other data base. Normally zero coefficients are not saved for SAGEM simulations. However it may help the LU decomposition in some models to save them. See also section 5.4.1 of GPD-2 for a further discussion of zero coefficients and similar options in GEMSIM and TABLO-generated programs.

7.6.2 GEMPIE Options

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

CPW Change page width
CPL Change page length

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

RPO Choose row print order

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

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

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

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

SNA Single solution not across the page

(Then the results for each component will be on a separate line.)

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

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

7.7 Model Name, Version and Identifier

The topics covered in this section are less important than when they were introduced. Now models are managed by managing their TABLO Input files rather than by specifying model names, versions and identifiers. So you can probably ignore this section.

When you create an Equations file, you must specify a **model name** (up to 8 characters), a **model version number** (an integer) and a **model identifier** (up to 60 characters). These are recorded on the Equations file and are also put on any Solution file produced from this Equations file. They then appear as a description of the model in simulation results (for example, on a GEMPIE Print file).

The model identifier only has this descriptive role, but the model name and version number are also used for checking, as described below.

If you use a closure stored on an Environment, Solution or LU file (see section 6.1.2 above), the model name and version number on this file are compared with those on the Equations file being used or created. If they are different a warning is issued, but you are still allowed to use the stored closure provided that the variables in the model, their names and numbers of components agree between the two files.

You may wish to use an LU file - that is, to take both the closure and LU decomposition of the left-hand-side matrix (see section 7.5 above) from this file, perhaps via a "use LU file ... ;" statement in a Command file. This will be allowed if the model name and version number on the LU file agree with those on the Equations file being used.[66] If they do not agree, you will only be permitted to take the closure from the LU file and the LU decomposition will be calculated from scratch.

Suggested Use

We suggest that you use the **model name** to reflect the original TABLO Input file (that is, to reflect the theory and structure of the model).

If you attach different base data for the same model, we suggest that you use a different **model version number**. (Changing base data changes the Equations file.) Then the checking above means that you will not inadvertently try to use the LU decomposition from one set of data with another set. (This would probably produce incorrect results.)

Different condensations of the same model result in different numbers of variables in the condensed version, and hence produce different Equations files. You may like to use slightly different model names to reflect these different condensations. (For example, use SJ for the model name of uncondensed Stylized Johansen but use SJC1 and SJC2 for the model names of two different condensations of this model.)

[66] You can only use an LU file (in the above sense) when you are using an existing Equations file. If you are creating a new Equations file via GEMSIM or a TABLO-generated program, you can take the closure from an old LU file but will not be allowed to read the LU decomposition itself from such a file.

CHAPTER 8

8. The Other GEMPACK Programs

This chapter is a guide to the other GEMPACK programs (that is, those apart from TABLO, GEMSIM, SAGEM, GEMPIE, MODHAR and TABLO-generated programs). Section 8.1 describes how you can use the program SUMEQ to obtain information about the Equations file, while section 8.2 describes the GEMPACK programs for data management. In section 8.3 we describe how you can use the program SLTOHT to convert simulation results into a form suitable for preparing reports (perhaps via non-GEMPACK utilities such as spreadsheets). In section 8.4 we explain how you can transfer models (including data and Solution files) to other machines. Section 8.5 outlines a couple of fairly specialised GEMPACK programs for comparing numerical results obtained on two different computers.

8.1 Information from Equations Files

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

$$\mathbf{Cz = 0} \qquad (1)$$

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

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

Part of the tableau for the 27 x 29 Equations Matrix C for Stylized Johansen is shown in Table 2.11.1 in section 2.11.1 above.

Notice that this divides up the Equations Matrix C into rectangular **submatrices**, where the columns and rows of each submatrix correspond respectively to components of the vector variable and the equations in the equation block determining this submatrix. Only some of these submatrices can contain nonzero entries. For example, the equation block "House" only involves the vector variables p_XH, p_Y and p_PC, so the submatrices in these rows corresponding to the other vector variables must be filled with zeros. Indeed, this is why, in general, the Equations Matrix of a general equilibrium model is sparse (as stated in section 7.5 above).

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

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

- It can produce a **map** of the Equations Matrix C, showing the variables and equations and the columns and rows in the Equations Matrix to which they correspond. This is described in more detail in section 8.1.1 below.
- It can produce information about row and column sums (across selected columns or rows, if required). This can be especially useful since it turns out to be a very good way of checking the implementation of many models, as described in more detail in section 8.1.2 below.
- You can use it to examine individual entries in the Equations Matrix. You specify the row and column and SUMEQ tells you the value of the entry in that row and column of the Equations Matrix. The map produced by SUMEQ (see section 8.1.1 below) will be helpful in deciding which rows and columns you are interested in.

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

8.1.1 Map of an Equations File Produced via SUMEQ

One of the options in the program SUMEQ is for a map of the Equations file. This shows

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

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

MAP OF VARIABLES AND EQUATIONS FOR STYLIZED JOHANSEN

```
  SETS
  ----

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

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

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

Map of Stylized Johansen Equations File Produced via SUMEQ

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

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

In order to identify the column corresponding to a particular component of a variable, you will also need to know the order of the components, which is documented in section 6.1.3 above. To identify the row corresponding to a particular equation, you will also need to know the order of the equations in a block, which is documented in section 4.10.5 of GPD-2.

8.1.2 Homogeneity Simulations using SUMEQ

This section introduces the topic of homogeneity simulations. More details, including a fully worked example, can be found in section 13.8 of GPD-4.

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

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

7-10, 30, 35-38

as an abbreviation for 7,8,9,10,30,35,36,37,38. [This is explained in much more detail in section 13.8 of GPD-4.]

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

each actual equation is in.) However, if you have condensed your model, it may be more difficult to spot which equation block is in error, since if an equation block with an error in it is substituted out, it may contaminate all equation blocks into which it is substituted.

8.2 Looking at or Comparing Data on Header Array Files

This section describes how you can get information about the data on a Header Array file. The relevant programs are

```
SUMHAR     SUMmarise a Header ARray file
SEEHAR     SEE the data on a Header ARray file
CMPHAR     CoMP the data on two Header ARray files
```

All 3 are easy to run interactively.

8.2.1 SUMHAR Summarizing a Header Array File

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

8.2.2 SEEHAR Prepare a Print File or CSV Spreadsheet File

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

1) If no special options are chosen, the default is to have row and column labels which indicate the position of each piece of data in the whole array.

2) An alternative is to have the output in a form which allows you to import the arrays into a spreadsheet easily.[67]
   The separator between the data entries can be chosen to be a comma (the default) or a TAB character (or any other desired character). To obtain this type of output, select the SEEHAR option

   SS Spreadsheet output

   at the start of the program, after which you will be asked which separator you want. Most spreadsheet programs (on PCs, for example) will accept as input the kind of file produced. If you choose a comma as the separator, the output will be in what these spreadsheet programs call **comma-separated values** or **CSV** format.

3) The third alternative is for the output to be a GEMPACK text data file, from which the data can be read easily into other GEMPACK programs. Specifically, each array is output in the form required for text input to MODHAR, GEMSIM or to TABLO-generated programs (as documented in chapter 20 of GPD-4). To obtain this type of output, select the SEEHAR option

   ROW Unlabelled output (row order)

   or

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

COL Unlabelled output (column order)

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

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

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

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

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

8.2.3 CMPHAR Comparing Data on Header Array Files

Run this program if you wish to compare the data at the same headers on 2 different Header Array files. The output is sent to a text file. For each header which is on both files, CMPHAR compares the data entry by entry if the arrays are of the same size (but ignores these arrays if they are of different size). Arrays associated with a header which is only on one of the files are ignored by CMPHAR.

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

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

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

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

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

CMPHAR Report

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

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

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

Significant Differences

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

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

Comparing Solution files

CMPHAR can also be used to compare the values on two Solution files from the same model. To do this, you must first convert the Solution files to Header Array files using the program SLTOHT (see section 8.3 below). Then select CMPHAR option

SOL Compare solution files

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

8.3 Processing and Reporting Simulation Results using SLTOHT

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

- to a Header Array file, or
- to a text file in one of several formats, including one which makes it easy to import the solution values into a spreadsheet program.[68]

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

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

> For example, if a Solution file from a set of Johansen simulations with Stylized Johansen contains 2 individual column solutions, one subtotal solution and a cumulative solution, making 4 solutions in all, SLTOHT will create
>
> - a 2 x 4 array for the variable 'p_XH' showing the values of its 2 components in these 4 different solutions,
> - a 1 x 4 array for the variable 'p_Y' showing the value of its only component in each of these 4 solutions,
> - a 2 x 2 x 4 array for variable 'p_XC' showing the values its 4 components in these different solutions (for example, the (1,2,3) entry in this array is the value of p_XC("s1","s2") in the third of these solutions, namely the subtotal solution),
>
> and so on.

Choosing solutions for output

After you have chosen the initial options, SLTOHT prompts for the Solution file name, reports the contents of the Solution file to the screen and presents a menu of possible solutions to be written to arrays. The range of choices depends on what SLTOHT finds on the Solution file.

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

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

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

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

Shocks and undefined solution values

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

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

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

8.3.1 Header Array file output

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

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

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

1. any individual solutions,
2. any subtotal solutions and
3. the cumulative solution. .

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

Header Mapping file

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

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

where <Header> is a 4-letter Header
<Variable Name> is up to 15 characters long
<No of Dims> is the number of dimensions for this variable
<Dims> is a list of the number of components in each dimension

The number of dimensions (No of Dims> and the list of dimensions <Dims> are optional and can be omitted
For example, using the default headers 0001, 0002,... the Header Mapping file for Stylized Johansen solutions is as follows:

Sample Header Mapping file for Stylized Johansen

```
0001 p_Y                  1      1
0002 p_PC                 1      2
0003 p_PF                 1      2
0004 p_XCOM               1      2
0005 p_XFAC               1      2
0006 p_XH                 1      2
0007 p_XC                 2      2      2
0008 p_XF                 2      2      2
0009 p_DVCOMIN            2      2      2
0010 p_DVFACIN            2      2      2
0011 p_DVHOUS             1      2
```

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

If you supply a Header Mapping file containing headers for only some of the variables, only solutions for these variables will be put on the Header Array file produced by SLTOHT. The easiest way of producing such a Mapping file is to edit one produced by SLTOHT.

If a Header Mapping file is not chosen, then SLTOHT will create one using a default set of Headers for the variables and using whatever dimensioning information is available on the Solution file to determine the variable dimensions. With Solution files that contain no variable or set information (such as all those created prior to GEMPACK Release 4.2.02), the best that SLTOHT can do is to assume that all variables are one-dimensional with size equal to the total number of components in the variable. The user can then edit this file and use it for subsequent runs of SLTOHT. For example, you can change the headers associated with each vector variable to ones more meaningful than the "0001" and so on used by SLTOHT as a default.

Alternative Header Array file output - Variable arguments ignored

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

VAI Variable arguments ignored (har output)

in the option choice at the start of SLTOHT.

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

For a variable z(i,j,k), the order of variable elements is with the first index changing most rapidly (see section 6.1.3). So the 24 components come in the order

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

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

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

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

8.3.2 Text file output, including spreadsheet output

To produce a text file, select from the options menu, presented when SLTOHT starts, one of the options below:

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

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

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

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

SS output optionally allows you to specify exactly which variables and which components you wish to output as described below under the heading "Spreadsheet Mapping File".

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

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

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

In each case for text file output, "how much data" information is output before the data values of the array. Details of the "how much data" information and structure of text files are described in chapter 20 of GPD-4.

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

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

With Release 6.0 of GEMPACK, SLTOHT has other options for spreadsheet output - see chapter 6 of GPD-4.

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

Sample of spreadsheet text output

```
! For each variable below columns represent the following solutions:
! 2 INDIVIDUAL column(s), each one giving the solution
!  for an individual shock.
! 1 SUB-TOTALS column(s) giving cumulative solutions
!  across sub-sets of the complete set of shocks.
! 1 TOTALS column, giving cumulative solutions across ALL shocks.
! For details about the shocks relevant to each column, run GEMPIE.
! ************************   SOLUTIONS   ************************
! Variable p_XH # Household demand for commodity i #
! Showing this as an array of size 2x4
! Solutions in columns, each row is a different component
2,4, real spreadsheet ;
     0.600000    ,   0.400000    ,    1.00000    ,    1.00000    ,
     0.700000    ,   0.300000    ,    1.00000    ,    1.00000    ,
! Variable p_XC # Intermediate inputs of commodity i to industry j #
! Showing this as an array of size 4x4
! Solutions in columns, each row is a different component
4,4, real spreadsheet ;
     0.600000    ,   0.400000    ,    1.00000    ,    1.00000    ,
     0.700000    ,   0.300000    ,    1.00000    ,    1.00000    ,
     0.600000    ,   0.400000    ,    1.00000    ,    1.00000    ,
     0.700000    ,   0.300000    ,    1.00000    ,    1.00000    ,
```

Spreadsheet Mapping Files

It is often useful to flexibly choose variables and components. In the spreadsheet option, you will be asked whether you want to use an existing SPREADSHEET MAPPING file. This file specifies the variables to be output and their components as shown below:

Sample Spreadsheet Mapping file for Stylized Johansen

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

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

Sample of spreadsheet output using mapping file

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

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

Syntax Rules for Spreadsheet Mapping File

The rules for a Spreadsheet Mapping file are as follows.

- Variable names must start in the first position in a line.
- Components are listed in one or more lines immediately following the variable name. All such lines must be indented at least one space to the right and the final line terminated with a semi-colon.
- Variables with no components specified will have all components output.
- Variables not specified will not appear on the output.
- Components may be repeated and may be in any order.
- Multidimensional arrays are treated as linear; thus p_XC(3) above refers to p_XC("s1","s2").
- Fatal errors occur if you don't indent, specify a component outside the range or leave out the semi-colon.

With Release 6.0 of GEMPACK, there are more possibilities for Spreadsheet Mapping files - see chapter 6 of GPD-4.

8.3.3 Description File

You can also elect, by choosing the option

DES Write description file

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

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

(b) If individual solutions are chosen, for each individual solution, the name and component number of the shock which gave rise to that solution. Each such individual description is terminated by a line beginning with '*'. The component number is reported as if the shocked variable were one dimensional.
(c) If subtotal solutions are chosen, for each subtotal solution, the description of that subtotal, as entered by the user in SAGEM, terminated by a line beginning with '%'. Then a list of the names (one name per line) of those variables of which there is at least one shocked component that contributed to this subtotal. Each such subtotal description is terminated by a line beginning with '*'.
(d) If the cumulative solution is on the Solution file, a list of the names (one name per line) of the variables of which there is at least one shocked component. The cumulative totals description is terminated by a line beginning with '*'.

8.4 Transferring Models Between Machines

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

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

Also helpful in using the model on a different machine are

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

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

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

If you are transferring to a machine with the same operating system (for example, from one DOS 80386 machine to another), you can probably just copy the relevant files to this machine.[69]

If, however, your target machine has a different operating system (for example, you are transferring from an 80386 DOS machine to an Apple Macintosh or to a mainframe), copying files will not succeed. GEMPACK contains several programs (namely MKHAR, RWHAR, MKEQ, RWEQ and SPJNTX) which make it easy to transfer models by following the procedures described below. The rest of this section is devoted to transferring between different operating systems.

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

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

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

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

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

8.4.1 Transferring the Whole Model

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

Step 1.

On Machine 1, convert any Header Array data files for the model to text files by running the GEMPACK program RWHAR. (Run it once for each such Header Array file. The program

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

requires just two inputs, firstly the name of the existing binary Header Array file and secondly the name of the text file to be created.)

Step 2.

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

Step 3.

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

Step 4.

Reimplement the model on Machine 2 by running TABLO etc.

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

```
                            Machine 1                Machine 2

TABLO Input file (text)     MODEL.TAB   ------->     MODEL.TAB

Header Array data file      MODEL.DAT
     (binary)                | (run RWHAR)
                             |
                           MODEL.HAF   -------->    MODEL.HAF
                                                     | (run MKHAR)
                                                     |
                                                    MODEL.DAT

Stored-input or            MODEL.STI    -------->    MODEL.STI
GEMPACK Command files      MODEL.CMF    -------->    MODEL.CMF
```

8.4.2 Transferring Johansen Simulation Capability

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

```
                            Machine 1                Machine 2

Equations file (binary)     MODEL.EQ4
                             | (run RWEQ)
                             |
                           MODEL.EQF   -------->    MODEL.EQF
                                                     | (run MKEQ)
                                                     | (run MKEQ)
                                                    MODEL.EQ4

Stored-input or            SIM1.STI     -------->    SIM1.STI
GEMPACK Command files      SIM2.CMF     -------->    SIM2.CMF
```

8.4.3 Transferring Solution Files

Unix users have asked us about using the Windows ViewSOL (see, for example, GPD-8) for looking at simulation results in Solution files. We have provided two programs **RWSOL** and **MKSOL** for transferring Solution files between machines. For example, to transfer a Solution file from a Unix machine to a windows PC, run RWSOL on the unix machine to convert the Solution file to a text file. Then transfer this text file (for example, using FTP) to the PC. Run MKSOL on the PC to convert this to a binary Solution file on the PC; you can use ViewSOL to look at this.

8.4.4 Handling Large Data Files

The data files for models can be quite large, which may present a problem in transferring the model. This is exacerbated for Header Array files since the text version (produced by RWHAR) is always much larger than the binary version.

Because of this, it may be necessary to split a text file into several parts before transferring it (perhaps to fit them onto disks if a PC is involved). This is what the GEMPACK program SPJNTX (**SP**lit or **J**oi**N T**e**X**t files) is designed to do.

For example, if you have a text file MODEL.HAF of size 4 megabytes (approximately 4 million characters), you can use SPJNTX on Machine 1 to split this into 6 files each of which will fit onto a 720K disk (that is, a disk holding about 720,000 characters). Then you can transfer these files (each on one disk) to Machine 2 and then run SPJNTX on Machine 2 to joint the files together again.

Splitting a File

Run SPJNTX, and select the default program options.

1. Respond 's' to the first prompt to indicate you want to split a file.
2. Respond '80' (the default) to the prompt asking the "maximum number of characters per line".
3. Your answer to the third prompt about the "maximum number of characters in each output file" will be determined by the size of output files you can handle. If, for example, you want each output file to fit onto one 720K disk, respond say '700000' (to allow a little safety margin from the maximum of about 720,000 characters per disk).
4. Respond 'n' (the default) to the prompt about omitting trailing blanks.
5. Give the name of the input file (the file to be split).
6. SPJNTX will produce as many output files as required, and will prompt you for the names of these one at a time. It will help you to put them back together if you give names which indicate their order, such as MODEL.1, then MODEL.2, and so on.

Joining Files

1. Respond 'j' to the first prompt to indicate you want to join files.
2. Respond '80' (the default) to the prompt asking the "maximum number of characters per line".
3. Respond 'n' (the default) to the prompt about omitting trailing blanks.
4. Give the name of the output file (the joined file).
5. When prompted, give the name of the next file to be added (for example, MODEL.1, MODEL.2, and so on). After the last one has been joined in, respond to the next prompt with a carriage-return to indicate that there are no more files.

8.5 Comparing Results from Different Machines

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

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

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

As with CMPHAR (see section 8.2.3 above) these programs report totals, averages and largest absolute differences and difference ratios.

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

CHAPTER 9

9. A Guide to the Rest of the Documentation

This document is the first of the GEMPACK documents most users should read.[70]

The other documents are

- a User's Guide to TABLO, GEMSIM and TABLO-generated programs (GPD-2),
- documentation of MODHAR (GPD-3),
- documentation of new features in Release 6.0 and changes from Releases 5.1 and 5.2 (GPD-4),
- machine specific documentation (GPD-6 for 386/486/pentium PCs running DOS, Windows 3.1, Windows 95 or Windows NT),
- documentation for installing and using the Executable-Image version (GPD-7), and
- document GPD-8 which suggests hands-on computing (on Windows machines or Unix and other machines working from a command prompt) to familiarise yourself with GEMPACK.

Users of GEMPACK on VAX/VMS, DEC Alpha/VMS and Unix machines should also be able to obtain machine-specific information from their GEMPACK Managers. (These are brief documents not included in the GPD series.)

9.1 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 (1994).

---

[70] The numbering of GEMPACK Documents has been re-started with Release 5 of GEMPACK, when the abbreviation "GPD" was first used. Previous editions of these documents did not have the same numbers as the current editions. Pre-Release-5 documents are numbered "GED-xx".

10. REFERENCES

Atkinson, Kendall E. (1989) *An Introduction to Numerical Analysis*, second edition, Wiley, New York.

Calder, Wayne (1992), 'Hitchhikers Guide to GEMPACK', *Impact Preliminary Working Paper* No. C10-01, Melbourne (October), pp.31.

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.

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.

11. GEMPACK DOCUMENTS[71]

Harrison, W.J. and K.R. Pearson (1998), *An Introduction to GEMPACK*, GEMPACK Document No. 1 **[GPD-1],** Monash University, Clayton, Fourth edition, October 1998.

Harrison, W.J. and K.R. Pearson (1994), *User's Guide to TABLO, GEMSIM and TABLO-generated Programs*, GEMPACK Document No. 2 **[GPD-2],** Monash University, Clayton, Second edition, April 1994.

Harrison, W.J. and K.R. Pearson (1993), *How to Create and Modify GEMPACK Header Array Files Using the Program MODHAR*, GEMPACK Document No. 3 **[GPD-3],** Monash University, Clayton, Third edition, April 1993.

Harrison, W.J. and K.R. Pearson (1998), *Release 6.0 of GEMPACK - New Features and Changes From Release 5.1 and 5.2*, GEMPACK Document No. 4 **[GPD-4],** Monash University, Clayton, Second edition, October 1998.

Harrison, W.J. and K.R. Pearson (1998), *Installing and Using the Source-Code Version of GEMPACK on DOS/Windows PCs with Lahey Fortran*, GEMPACK Document No. 6 **[GPD-6],** Monash University, Clayton, Eighth edition, October 1998.

Harrison, W.J. and K.R. Pearson (1998), *Installing and Using the Executable-Image Version of GEMPACK on DOS/Windows PCs*, GEMPACK Document No. 7 **[GPD-7],** Monash University, Clayton, Fifth edition, October 1998.

Harrison, W.J. and K.R. Pearson (1998), *Getting Started with GEMPACK: Hands-on Examples*, GEMPACK Document No. 8 **[GPD-8],** Monash University, Clayton, First edition, October 1998.

[71] 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".

—A—

Absolute difference, 8-151
Accuracy, 2-51, 7-143
Actions
  GEMSIM and TABLO-generated program, 4-101
Address
  Contact, 1-9
ASCII file, 5-118
Auxiliary Statement, Table files, 4-107
Auxiliary Statement,Table files, 2-30

—B—

Backsolving, 3-97, 3-98
Batch operation, 5-110, 5-112
Binary files
  Transfer of files, 8-160
  Use in GEMPACK, 5-118
Bug
  Reporting, 5-115

—C—

C matrix, 2-49, 7-142
  Condensation, 3-95
Calibration, 3-89
Case sensitive, 3-86, 5-109
  TABLO, 3-67
Change UPDATE, 3-87
Change variables, 3-69
Check
  TABLO, 3-95, 3-99
Checking models, 8-149
Choice of variables
  Command file, 6-127
  Interactive, 6-131
  Macros,all,1, 6-132
  Sets, 6-133
  To shock or print, 6-127
Closure, 2-30, 2-33, 6-121
  Different shocks, 7-142
  Meaning, 6-121
  Modifying commands, 6-125
  Saving, 6-124
  Specifying, 6-123
  Valid, 6-121
CMPHAR, 8-151
Code
  TABLO, 3-95, 3-99
COEFFICIENT
  TABLO, 3-68
Column order
  SLTOHT, 8-156
Column sums
  Equations, 8-148
Columns
  Solution, 7-137
Command file, 2-30
  Closure, 6-123
  Eliminating syntax errors, 5-116
  SAGEM, 6-129
Command file syntax
  Choice of variables, 6-127
  Shocks, 6-134
Comments, 5-109
  Text files, 3-73
Comparing
  Header Arrays, 8-151
  Solution files, 8-151
Compile, 2-30
Computation time
  Extrapolation, 2-52
  TABLO, 3-99
Condensation, 3-95
  Example, 3-96
Condense
  TABLO, 3-95, 3-99
Condensing twice, 3-99
Contact
  GEMPACK, 1-9
CPU time
  SAGEM, 7-144
CSV format, 3-77, 3-78, 8-150
Cumulative results, 2-47, 7-138
Cumulative solution, 8-154
Cumulatively-retained
  Choice of, 6-127
Cumulatively-retained endogenous
  Backsolves, 3-98
  Meaning, 7-138

—D—

Data
  Comparing, 8-150
  Post-simulation, 2-41
  Pre-simulation, 2-41
  Updated, 2-41
Data file, 3-71, 5-118
Data manipulation, 8-153
  Actions, 4-102
  TABLO, 3-76
Default response, 5-109
Demonstration Version, 1-3, 1-4
Difference ratio, 8-151
Different machines, 1-1
  Binary compatible, 8-160
DISPLAY
  TABLO, 3-68
Documentation, 9-165
DPPW
  Reference, 2-12

—E—

Elasticity, 7-137
Eliminated variables, 3-98
Endogenous, 2-11, 6-121
  Individual, cumulative, 7-138
Environment
  Meaning, 6-121
Environment file, 5-119

EQUATION
TABLO, 3-61, 3-68
Equation block, 2-49
EQUATION(NONE) statement, 3-77
Equations, 2-49
Linearized, 2-11
Solving, 7-143
Equations file
Contents, 2-49, 8-147
Creation, 4-101
Production and use, 5-119
Equations matrix, 2-49, 7-142
Condensation, 3-95
Map, 8-147
Error message, 5-114
Errors in Command file
Eliminating, 5-116
Errors rounding, 8-152
Euler's method, 2-51, 4-103
Exclamation mark, 5-109
Executable image, 2-14
Executable-image licence limits, 1-3
Executable-image licences, 1-3
Limited, 1-3
Unlimited, 1-4
Executable-image version
Limited, 1-3
Unlimited, 1-4
Executable-image Versions, 1-3
Exogenous, 2-11, 6-121
Individual, cumulative, 7-138
Extrapolation, 2-51, 4-103
Extrapolation Accuracy file, 2-40, 5-119
Contents, 2-52
Extrapolation Accuracy Summary, 2-53

—F—

Fatal Error, 5-114
FILE
Logical name, 3-68
TABLO, 3-68
FORMULA & EQUATION, 3-68
Fortran compiler, 1-3
FTP, 3-78, 8-160

—G—

GEMPACK
Versions, 1-3
World Wide Web, 1-9
GEMPACK Documents, 9-165
GEMPACK licences, 1-3
Executable-image licence limits, 1-3
GEMPACK programs, 1-2
GEMPACK-L
Mailing List, 1-9
GEMPIE
Choice of variables, 6-127
Example, 2-31, 2-33
Options, 7-144
Print file, 2-14, 2-16, 2-31, 7-139
GEMSIM
How to run, 2-33
Options menu, 4-102
GEMSIM Auxiliary files
Meaning, 2-16
GEMSIM Statement, Table files, 2-32
Gragg's method, 2-51, 4-103

—H—

Harwell Laboratories, 7-143
Harwell parameter, 7-143
Header, 3-77
Header Array
Transfer, 8-160
Header Array file, 8-150
From Solution, 8-153
Header Array files, 3-77
Creating, 3-71
Modifying, 3-71
Header Mapping file
SLTOHT, 8-154
Headers
Reserved, 3-77
Help
Online, 5-114
Homogeneity, 8-149
How much data information, 3-71

—I—

Ill-conditioned matrix, 7-144
Impact Project, 1-9
Implementation, 2-11, 3-99
Increasing Parameters, 5-114
Individual column results, 2-45, 7-137
Printing, 7-141
Individual solutions, 8-154
Individually-retained, 7-137
Choice of, 6-127
Information file, 2-30, 2-32
Interactive operation, 5-110, 5-112
Internal program error, 5-115
Invalid input
ASI, SIF, BAT, 5-112
Iterative refinement, 7-143

—J—

Johansen simulation, 2-42, 2-50, 7-137
Johansen solution
Meaning, 2-50
Johansen's method, 4-103

—K—

Kermit, 3-78, 8-160
Keyword, 3-61, 3-67

—L—

Labelling information, 3-67
Large data files
Transferring, 8-162
Large Models
Condensation, 3-95

Large-simulations licence, 1-4
When required, 1-4
Levels equations, 2-12, 3-57
Levels variables, 2-12, 3-69
Licence
Large-simulations, 1-4
Licences
Executable-image licence limits, 1-3
Licences for GEMPACK, 1-3
Limited Executable-image version, 1-3
Limited Executable-image version licences, 1-3
Linearized equations, 3-57
Linerarized equations, 2-49
Lists
Choice of variables, 6-127
Log file, 2-40, 5-110
Terminal output, 5-112
Longname
Header Array file, 3-77
LU decomposition, 4-105, 7-143

—M—

MA28, 7-143
Macro variable
Meaning, 2-19
Macro variables, 6-128, 6-131
Mailing List
GEMPACK-L, 1-9
Map of Equations file, 8-148
Mapping file
Header, 8-154
SLTOHT Header Array, 8-154
Spreadsheet, 8-157
Matrix
Equations, 7-142
Ill-conditioned, 7-144
Sparse, 2-50
Memory
TABLO-generated programs, 4-108
Midpoint method, 2-51
Miniature ORANI
Example, 6-122
MKEQ, 8-160
MKHAR, 8-160
Model examples, 1-6
Model identifier, 7-146
Model name, 7-146
Model Representation
Levels,Mixed,Linear, 3-93
Model users, 1-5
Model verification, 8-149
Model version number, 7-146
MODHAR, 3-75, 3-77
Modifying a closure, 6-125
Multi-step simulation, 4-103
Multi-step solution, 2-49, 2-51
No subtotals, 7-142

—N—

Non-linear equation, 2-51
Numerical values
Shocks, 6-134

—O—

Omitting variables, 3-98
Online Help, 5-114
Options
CMPHAR SOL, 8-152
GEMPACK BPR, 5-113
GEMPACK Basic, 5-110
GEMPACK BAT, 5-111
GEMPACK LOG, 5-111, 5-113
GEMPACK SIF, ASI, 5-111
GEMPIE CPL, 7-145
GEMPIE CPW, 7-145
GEMPIE FBS, 7-145
GEMPIE NBV, 7-145
GEMPIE OSR, 7-145
GEMPIE OSS, 7-145
GEMPIE RPO, 7-145
GEMPIE SNA, 7-145
SAGEM CMF, 7-144
SAGEM CPU, 7-144
SAGEM KZC, 7-145
SAGEM NIR, 7-144
SAGEM NSM, 7-144
SAGEM NWE, 7-144
SEEHAR COL, 8-150
SEEHAR NEL, 8-151
SEEHAR ROW, 8-150
SEEHAR SS, 8-150
SLTOHT VAI, 8-155
TABLO Code PGS, 2-16
TABLO Code WFP, 2-14
Options menu
GEMPACK programs, 5-110
TABLO-generated program or GEMSIM, 4-102
Outputs from SAGEM, 7-137

—P—

Page width,length
GEMPIE, 7-145
Parameters
In equations, 3-58
Program, 5-114
Passes
TABLO-generated program or GEMSIM, 4-103
Percentage change, 2-12
Percentage change variables, 3-69
Post-simulation data, 2-41
Pre-simulation data, 2-41
Print file
Meaning, 5-118
Program error, 5-115
Prompts
Choice of variables, 6-127

—Q—

Quantifier, 3-62
Quantifier declaration, 3-67

—R—

READ

TABLO, 3-68
Reading data, 3-71, 5-118
Reducing model size
Condensation, 3-95
Repeated values, 6-136
Reporting program errors, 5-115
Residual correction, 7-143
Rounding error, 7-143
Row order
SLTOHT, 8-156
Row print order
GEMPIE, 7-145
Row sums
Equations, 8-148
Row totals, 7-138
RunGEM, 1-5
RWEQ, 8-160
RWHAR, 8-160

—S—

SAGEM, 2-42
Backsolved variables, 3-98
Choice of variables, 6-127
Johansen simulation, 7-137
Many solutions, 7-142
Solution file, 7-138
Subtotal, 7-138
SAGEM Options, 7-144
Scalar variable
Meaning, 2-19
Scalar variables, 6-128
SEEHAR, 3-77, 3-78, 8-150
SET
TABLO, 3-67
Set and element labelling, 3-74
Sets of variables
Choosing, 6-133
Shock files, 6-136
Shocks
For one step, 4-106
Meaning, 6-121
Multi-dimensional variable, 6-136
Numerical values, 6-134
Same closure, 7-142
SLTOHT, 8-154
Specifying, 6-134
Shocks - Uniform
Meaning, 6-134
Significant difference, 8-152
Simulation
Choice of variables, 6-127
Meaning, 2-11
SAGEM, 7-137
Steps, 2-13
Stylized Johansen example, 2-18
Simulations
Several at once, 2-42
SLTOHT, 3-78, 8-153
Spreadsheet Mapping file, 8-157
Solution
Computation time, 2-52
Matrix, 7-137
Simulation, 7-137
Subtotal, 7-138
To Header Array, 8-153
To Text file, 8-153
Solution file, 2-14, 2-16, 2-31
Solution files
Contents, 7-137
SAGEM, 7-137
Transferring between machines, 8-162
Solving models, 2-11
Source-code versions, 1-3
Sparse matrix, 7-143, 8-147
SPJNTX, 8-160, 8-162
Spreadsheet, 3-77, 8-150
Example, 8-156, 8-157
From Solution, 8-153
Separator, 3-78
SLTOHT, 8-156
Spreadsheet Mapping file, 8-157
Statements in TABLO Input file, 3-67
Steps
TABLO-generated program or GEMSIM, 4-103
Stored-input file, 5-110
Terminal output, 5-112
Stylized Johansen model
Condensation, 3-96
Data, 3-60
Description, 2-18
Equations, 3-57
Equations matrix C, 2-49
GEMSIM simulation, 2-33
SAGEM simulation, 2-44
TABLO Input file, 3-64
TABLO-generated simulation, 2-30
Submatrix, 8-147
Subset
Choice of variables, 6-127
Substitution, 3-95, 3-98
Subtotal, 2-47, 7-139
Command file syntax, 6-127
Meaning of, 7-138
Printing, 7-141
Subtotal solution, 8-154
SUM
TABLO, 3-68
SUMEQ, 8-147
SUMHAR, 3-77, 8-150
Swapping variables, 6-125
Syntax errors in Command file
Eliminating, 5-116

—T—

Tableau
Equations matrix C, 2-49
TABLO
How to run, 2-29, 2-32
Record, Table files, 3-99
Restarting, 3-99
Stages, 3-99
TABLO Input file, 2-11, 3-55, 3-61
Comment, 3-67
Data manipulation, 3-76
Levels,Mixed,Linear, 3-93
Statements, 3-67

TABLO language, 3-70
TABLO stages, 3-95
TABLO Syntax
  Example, 3-61
TABLO-generated program
  Actions, 4-101
  Meaning, 2-14
  Options menu, 4-102
TABLO-generated program or GEMSIM
  Choice of variables, 6-127
TABLO-generated programs
  Memory required, 4-108
Terminal input
  Comments, 5-109
Terminal output
  Suppressing, 5-112
Text files, 3-77
  From Solutions, 8-153
  Input to MODHAR, 3-72
  Transfer of files, 8-160
  Use in GEMPACK, 5-118
Transferring data
  Between PCa and other computers, 8-160
  Between PCs and mainframes, 3-78
Transferring models, 8-160
Transferring Solution files to PCs, 8-162

—U—

U parameter, 7-143
Undefined Solution values
  SLTOHT, 8-154
Unlimited Executable-image version, 1-4
Unlimited Executable-image version licences, 1-4
UPDATE
  Actions, 4-101
  Example, 3-87
  Meaning, 3-87
  Product, 3-87
  Purpose, 3-80
UPDATE(CHANGE), 3-87
Updated data, 2-41
  After Johansen simulation, 2-47
Updated data file, 5-119
User Input
  Condensation in TABLO, 3-96
  GEMPIE Example, 2-31, 2-33
  MODHAR Modify data, 3-75
  TABL condense and Code, 3-100
  TABLO - Step 1, 2-29
  TABLO Check only, 3-100

—V—

VARIABLE
  TABLO, 3-67
Variables, 2-49
  Change, 3-69
  Levels, 2-12
  Percentage change, 2-12, 3-69
  Shocked, 6-127
Vector variable
  Meaning, 2-19
Vector variables, 2-49
Verbal description, 2-39
Versions of GEMPACK, 1-3
ViewHAR, 3-77
ViewSOL, 8-162

—W—

WinGEM, 1-5
World Wide Web, 1-9
WRITE
  TABLO, 3-68

—Z—

Zero coefficients, 7-145