CENTRE of POLICY STUDIES and the IMPACT PROJECT

Eleventh Floor
Menzies Building
Monash University Wellington Road
CLAYTON Vic 3168 AUSTRALIA

Telephone: from overseas:
(03) 905 2398, (03) 905 5112 61 3 905 2398 or 61 3 905 5112
Fax numbers: from overseas:
(03) 905 2426, (03)905 5486 61 3 905 2426 or 61 3 905 5486
e-mail impact@vaxc.cc.monash.edu.au

Computing Solutions for Large General Equilibrium Models Using GEMPACK

by

W.Jill Harrison and K.R. Pearson

Impact Project
Monash University

Preliminary Working Paper No. IP-64 June 1994

ISSN 1031 9034 ISBN 0 7326 0716 7

ABSTRACT

GEMPACK 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 a versatile method for solving intertemporal models. GEMPACK provides software for calculating accurate solutions of an economic model, starting from an algebraic representation of the equations of the model. These equations can be written as levels equations, linearized equations or a mixture of these two. The software provides a range of utility programs for handling the economic data base and the results of simulations, and is fully documented from a user's point of view.

GEMPACK is used to implement and solve a number of economic models including several single-country models (of which the ORANI model of Australia is perhaps the best known), multi-country trade models, regional models and intertemporal (or dynamic) models. GEMPACK runs on a wide variety of computers including 80386/80486 microcomputers running DOS, Windows or OS/2, Apple Macintosh computers, Unix machines, DEC VAX and Alpha machines running VMS.

This paper gives an overview of the current release of GEMPACK (Release 5.1, April 1994). Included are descriptions of

- the algebra-like language used to describe and document the equations of an economic model,
- the operation of the pre-processor TABLO which converts the equations of the model to a form suitable for computing solutions of the model,
- the solution methods used for producing accurate solutions of the model,
- the facilities for specifying and carrying out simulations, including the options for varying the choice of endogenous and exogenous variables and the variables shocked,
- the condensation facility which makes it possible to solve very large models,
- the utility programs for assisting in managing the data bases on which models are based,
- the different versions of GEMPACK.

CONTENTS

		page
Abstract		*i*
1	*INTRODUCTION*	*1*
	1.1 Development of GEMPACK	*1*
	1.2 Software For Equilibrium Models	*2*
	1.3 Overview Of The Paper	*3*
	1.4 Simulations	*4*
2	*IMPLEMENTING MODELS*	*6*
	2.1 Writing Down The Equations Of A Model	*6*
	2.1.1 The Equations of Stylized Johansen	*8*
	2.2 Data Requirements For A Model	*9*
	2.2.1 The Data Requirements For Stylized Johansen	*10*
	2.3 Constructing The TABLO Input File For A Model	*10*
	2.3.1 The TABLO Input File For Stylized Johansen	*11*
	2.4 Linearized Representations And Update Statements	*13*
	2.5 Different Representations (Levels, Linearized or Mixed)	*15*
	2.5.1 TABLO Linearizes Levels Equations	*16*
	2.6 Example Models	*16*
3	*CARRYING OUT SIMULATIONS*	*17*
	3.1 Interpreting The Results of A Simulation	*17*
	3.1.1 A Simulation With Stylized Johansen	*17*
	3.2 Specifying A Simulation	*19*
	3.2.1 An Example Command File	*20*
	3.3 Steps In Carrying Out A Simulation	*22*
	3.4 Different Closures And Shocks	*24*
	3.4.1 Different Closures	*25*
	3.4.2 Shocks	*26*
4	*HOW GEMPACK SOLVES THE EQUATIONS*	*26*
	4.1 Johansen Solutions	*26*
	4.2 Multi-Step Solutions	*27*
	4.3 Solution Methods And Extrapolation	*28*
	4.3.1 Connection With Initial Value Problems	*30*
	4.4 Several Johansen Simulations At Once	*30*
5	*CONDENSING LARGE MODELS*	*32*
	5.1 Substituting Out Variables	*32*
	5.2 Backsolving for Variables	*33*
	5.3 Omitting Variables	*34*
	5.4 The Effect Of Substitutions On Computational Complexity	*34*

6 DATA PREPARATION AND RESULT REPORTING

6.1 Data Preparation 35
6.2 Result Processing And Reporting 36

7 INTERTEMPORAL MODELS 36

8 COMMUNICATING MODELS TO OTHERS 37

9 SOFTWARE ASPECTS 37

9.1 Standard Program Options 37
9.2 Subroutines 38
9.3 Model Size And Program Parameters 38
9.4 History Of Files 39
9.5 Binary Files 39
9.6 No Special Windows Features 39
9.7 Solving Sparse Systems Of Linear Equations 40
9.8 Compilers On 386/486 PCs And Macintosh PCs 40

10 CHANGES IN COMPUTING ENVIRONMENT 40

11 DIFFERENT VERSIONS OF GEMPACK

11.1 Source-code Versions 41
11.2 Executable Image Version 41
11.3 Demonstration Version 41
11.4 Current GEMPACK User Documentation 42

APPENDIX A THE TABLO INPUT FILE FOR THE STYLIZED JOHANSEN MODEL 42

APPENDIX B INITIAL VALUE PROBLEMS 47

B.1 Simulation Problems - A Definition 48
B.2 Converting a Simulation Problem To An Initial Value Problem 49
B.3 Setting up the Initial Value Problem 52

REFERENCES 53

LIST OF TABLES

Table 2.1.1a	*Levels And Linearized Equations of the Stylized Johansen Model*	*7*
Table 2.1.1b	*Levels Variables for Stylized Johansen*	*8*
Table 2.1.1c	*Parameters for Stylized Johansen*	*9*
Table 2.1.1d	*Equations for Stylized Johansen*	*9*
Table 3.1.1a	*Input-output Data Base for Stylized Johansen*	*18*
Table 3.1.1b	*Part of Simulation Results File*	*18*
Table 4.3	*Multistep and Extrapolated Results*	*29*
Table 4.4	*Individual Column Results for Johansen Simulations*	*31*

LIST OF FIGURES

Figure 1.4a	*Comparative-Static Interpretation of Results*	*5*
Figure 1.4b	*Forecasting Interpretation of Results*	*5*
Figure 2.4	*TABLO Statements for one Equation*	*14*
Figure 2.5a	*Levels TABLO Input file*	*15*
Figure 2.5b	*Linearized TABLO Input file*	*15*
Figure 3.2	*The Information Required to Specify a Simulation*	*20*
Figure 3.2.1	*Example of a GEMPACK Command File*	*21*
Figure 3.3	*Steps in carrying out a Simulation*	*23*
Figure 4.2	*Multi-step solution using Euler's method*	*28*

COMPUTING SOLUTIONS FOR LARGE GENERAL EQUILIBRIUM MODELS USING GEMPACK

by

W. Jill HARRISON and K.R. PEARSON

1 INTRODUCTION

GEMPACK 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 a versatile method for solving intertemporal models.

GEMPACK software is being used in over 50 organizations around the world (universities, government departments and private sector firms). It is used to implement and solve a number of economic models including several single-country models (of which the ORANI model of Australia is perhaps the best known), multi-country trade models, regional models and intertemporal (or dynamic) models.

GEMPACK provides software for calculating accurate solutions of an economic model, starting from an algebraic representation of the equations of the model. These equations can be written as levels equations, linearized equations or a mixture of these two.

The software provides a range of utility programs for handling the economic data base and the results of simulations, and is fully documented from a user's point of view.

GEMPACK runs on a wide variety of computers including

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

1.1 Development Of GEMPACK

The aim in developing GEMPACK has been to provide a suite of tools (or, expressed more exotically, a modelling environment) for equilibrium modellers which will free them from most computing-related difficulties and constraints, and will allow them to concentrate on the economic aspects of their models. In particular, modellers using GEMPACK should never have to write their own programs either to solve their model or to communicate it (theory or data) to the computer or to other modellers. The algebraic representation of models used in GEMPACK (see section 2 for more details) has been chosen and designed with the intention that

(a) modellers will find it relatively easy to write down and/or modify the theory of their models,

(b) it should be an intelligible, essentially self-contained, documentation of the model, and

(c) it should be the means of communicating the model to others who wish to understand, use and/or modify the model.

An important part of this was the desire to provide a tool that would reduce by an order of magnitude the research resources (especially person-months) required to build and maintain a new model. It has been estimated that, compared to the situation which was common in the middle 80s when modellers often wrote their own programs (for example, in Fortran), the use of GEMPACK reduces this research input by over 85% - see Powell (1988).

The rationale behind the development of GEMPACK is still essentially as set out in Pearson (1988). An introduction to the algebraic language used by GEMPACK can be found in Codsi and Pearson(1988); however that paper was written before GEMPACK was able to produce accurate solutions of the (usually nonlinear) equations of a model, and before GEMPACK allowed explicit levels equations. An introduction to the intertemporal capabilities of GEMPACK can be found in Codsi, Pearson and Wilcoxen (1992).

The software is general-purpose in the sense that it can be used to model a wide range of economic behaviour. It imposes no fixed recipe of possible behaviours; rather it allows most sorts of behaviour that can be expressed as algebraic equations. The software "knows" no economics; it is the modeller's responsibility to ensure that the algebraic equations are accurate. The algebraic language does not allow explicit inequalities in the system solved. However, because linearized equations are allowed, various sorts of optimising behaviour whose explicit solution in the levels may be complicated or not known analytically can be handled; the first-order conditions can be used instead of the optimising form of the problem.

1.2 Software for Equilibrium Models

There are various general-purpose software packages available for solving equilibrium models. Probably the best known is the fine GAMS software (see Brooke el al (1988)), which now incorporates MPS/GE (see Rutherford (1989)). There is a large overlap in the sorts of models that can be handled by GAMS, MPS/GE and GEMPACK. There are many similarities between these. For example, the algebraic languages used by GAMS and GEMPACK are similar; the development of this interface for GEMPACK benefited from our knowledge of the GAMS language. There are also significant differences between these packages. A recent review of some software for equilibrium modelling (including GEMPACK and MPS/GE, but not GAMS) can be found in Harrigan (1993).

It is not our intention here to compare GEMPACK with other packages (we leave this to others), but rather to give an up-to-date description of the main features of GEMPACK (as in Release 5.1). We believe that the existence of overlapping but different general-purpose tools for GE modelling has proved, and will continue to prove, a stimulus for such modelling. Modellers are free to choose the tool which is best suited to the task before them, or with which they are most familiar.

Of course it is important that the different software packages produce the same solution to a modelling simulation, irrespective of the representation used to communicate the model to the computer and of the algorithm used to solve the

model. Hertel *et al.* (1992) address this issue and explain that, despite the linear representation often used to communicate models to GEMPACK, GEMPACK can produce the same accurate solutions of the underlying levels equations of the model as are produced by systems such as GAMS which usually begin from explicit levels equations. More details can be found in section 4 below.

Features of GEMPACK which, so far as we are aware, are not explicitly available in any other software are

- automatic algebraic condensation of a model (see section 5). (This makes it possible to solve much larger models than would otherwise be feasible.)
- the acceptance of wholly linearized equations or a mixture of levels and linearized equations. (See Harrison *et al.* (1993) and section 2 below for more details.)
- software for computing several Johansen solutions simultaneously (see section 4.4).

1.3 Overview Of The Paper

This paper gives an overview of the current release of GEMPACK (Release 5.1, April 1994).

The algebra-like language used to describe and document the equations of an economic model is described in sections 2.1 and 2.3. The pre-processor program TABLO converts the equations of the model to a form suitable for carrying out simulations and computing solutions of the model. The equations, which can be (non-linear) levels equations or linearized equations or a mixture of the two, are always converted to a linearized representation of the model; any levels equations are symbolically differentiated by the software (see sections 2.4 to 2.7).

In section 3, we describe the facilities for specifying and carrying out simulations, including the options for varying the choice of endogenous and exogenous variables and the variables shocked. These include GEMPACK Command files which consist of a number of self-documenting statements such as

```
exogenous to txs tms tm tx qo(ENDW_COMM,REG) ;
rest endogenous ;
shock tms("food","USA","EEC") = -10.0 ;
```

The solution methods used for producing accurate solutions of the model are given in section 4. These multi-step methods, which are based on the linearized representation produced by TABLO, are variants of well-known numerical methods for solving initial-value problems involving differential equations.

The condensation facility in GEMPACK, which makes it possible to solve very large models, is described in section 5. The software can be directed to make algebraic substitutions symbolically in the system of linearized equations to reduce the system actually solved to a manageable size. The values of those variables which are substituted out are available, if desired.

Data requirements for models are illustrated in section 2.2, and the utility programs for assisting in managing the data bases on which models are based are discussed in section 6. Data can be held in binary or text form. The data can be inspected, modified, converted to spreadsheets or moved to different machines

(including those with different operating systems). The GEMPACK system of communicating models to other modellers (and other machines) is described in section 8.

A brief introduction to the method GEMPACK uses for solving intertemporal models is given in section 7. In section 9, we describe various aspects of the software design. Changes in the computing environment, discussed in section 10, have caused a move to the use of PCs instead of mainframes for many users.

Finally, details are given in section 11 of the different versions of GEMPACK. Most are source-code versions, which require a suitable Fortran compiler; for these the size of models that can be handled is limited only by the amount of memory available. Others are executable-image versions, which can only handle models of limited size. There is a Demonstration Version available essentially free for use in teaching or for modellers wanting to find out more about how the software works.

1.4 Simulations

"Simulation" can mean different things in different contexts. Here we describe what it typically means in a general equilibrium model setting.

Solving models within GEMPACK is always done in the context of a simulation. The values of certain of the variables, the exogenous variables, are specified and the software calculates the values of the remaining ones, the endogenous variables. The new values of the exogenous variables are usually given by specifying the percentage changes (increases or decreases) from their values in the original (pre-simulation) solution. Similarly the results of the simulation, the endogenous variables, are usually reported as percentage changes.

General equilibrium models were first used to give policy advice. More recently, some have been used for forecasting. Below we follow Horridge *et al.* (1993) in giving a brief explanation of these.

Many policy-advice simulations are the answer to "What if" questions such as "If tariffs are reduced by 10 percent on a range of commodities, how much different would the economy be in 5 years time from what *it would otherwise have been*?" As shown in Figure 1.4a we think of the initial (pre-simulation) solution and data base as representing the state of the economy as it would be in (say) 5 years' time with no tariff change. The new (post-simulation) solution represents the state of the economy as it would be in 5 years' time with reduced tariffs but no other policy changes. For employment, say, A might be its value now, B its value in 5 years' time with no tariff change and C its value in 5 years after the tariff reduction. Then the result reported by GEMPACK would be the percentage change from B to C, namely 100(C-B)/B. This is often called the **comparative-static** interpretation of results.

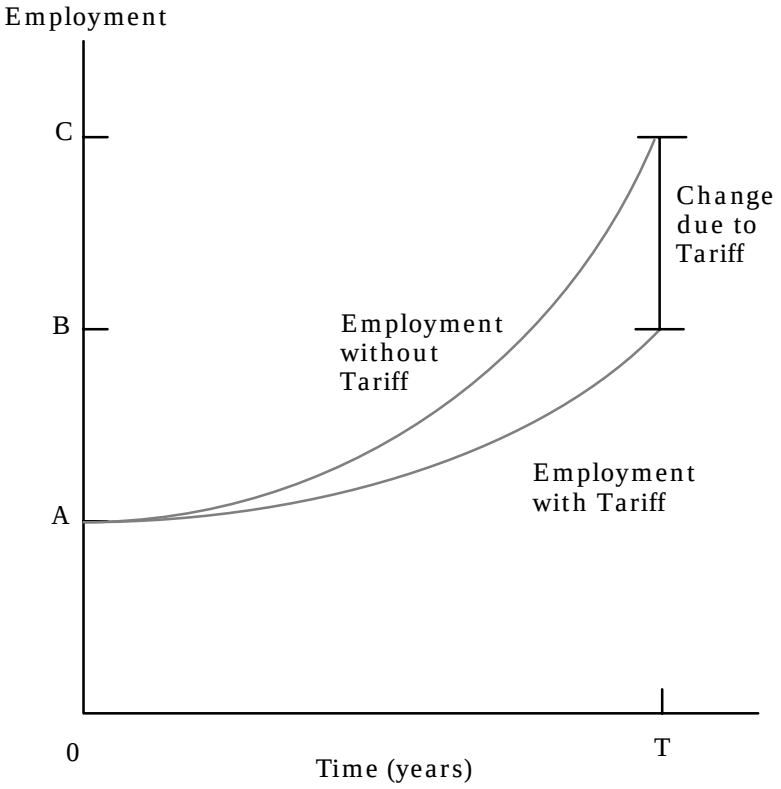

Figure 1.4a: Comparative-Static Interpretation of Results

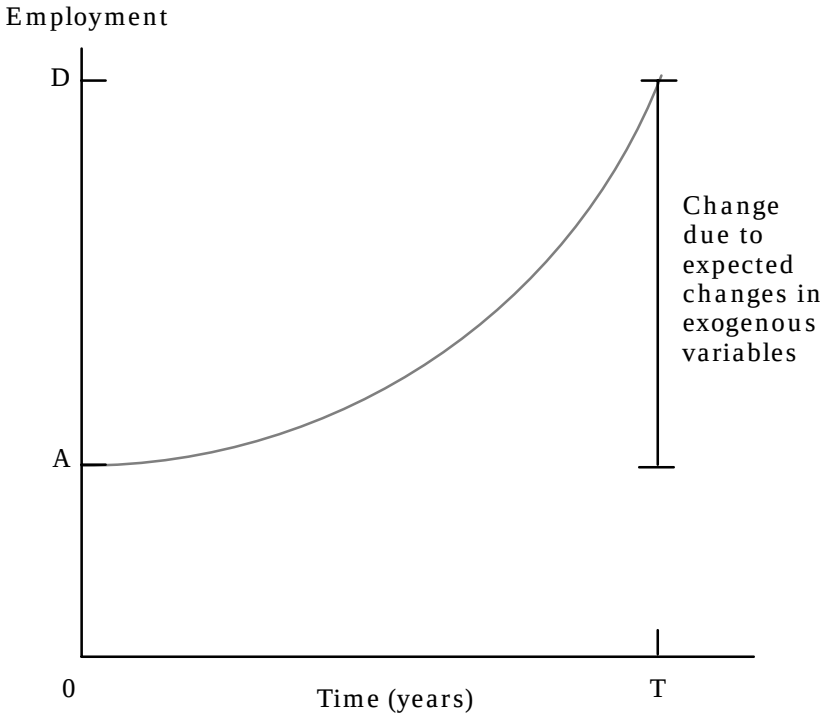

Figure 1.4b: Forecasting Interpretation of Results

For forecasting, it is necessary to feed in (as shocks to the model) expected changes in all exogenous variables over the time span of the simulation; the model should then report changes in the endogenous variables. In Figure 1.4b, the initial (pre-simulation) solution and data base are thought of as representing the economy now, and the final (post-simulation) solution and data base as those that will be in 5 years' time. The results reported by GEMPACK are percentage changes over the period. For example, if employment is A now and will be D in 5 years' time (given the expected changes in the exogenous variables), the result reported will be the percentage change 100(D-A)/A from A to D.

Typically only a small number of exogenous variables are shocked in policy-advice simulations but a large number are shocked in forecasting simulations. We look at simulations in greater detail in section 3.

2 IMPLEMENTING MODELS

A model is implemented in GEMPACK when

(1) the equations describing its economic behaviour are written down in an algebraic form,

(2) data describing one solution of the model are assembled, to be used as a starting point for simulations, and

(3) a text file, containing the equations (written in an algebra-like syntax) and information about the data, is prepared. This file is called a **TABLO Input file** since TABLO is the name of the GEMPACK program which processes this file and converts the information on it to a form suitable for running simulations on the model.

These three stages are described in sections 2.1 to 2.3 respectively. We illustrate the process by showing, in sections 2.1.1, 2.2.1 and 2.3.1, how these stages are carried out for the Stylized Johansen model. This small model, which is used for teaching purposes, is described in Chapter 3 of Dixon *et al.* (1992).

In sections 2.4 and 2.5 we discuss briefly the differences between linearized, levels and mixed representations of a model. Section 2.6 contains references to models implemented via GEMPACK and those usually supplied with GEMPACK.

2.1 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 the equations of the model are written down in ordinary algebra, it is a simple matter to put them into a TABLO Input file.

Levels or linearized versions of the equations can be used, 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

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

Table 2.1.1a: Levels and Linearized Equations of the Stylized Johansen Model*

Levels Form	Linearized Form	
consumer demands $X_{i0} = \alpha_{i0}\, Y / P_i$	$x_{i0} = y - p_i$	i = 1,2
intermediate demands $X_{ij} = \alpha_{ij}\, X_j \prod_{t=1}^{4} P_t^{\alpha_{tj}} \left[\prod_{t=1}^{4} (\alpha_{tj})^{-\alpha_{tj}} \right] / [A_j\, P_i]$	$x_{ij} = x_j - (p_i - \sum_{t=1}^{4} \alpha_{tj}\, p_t)$	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.

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 is included, GEMPACK can still produce accurate solutions of the underlying levels equations (which are usually nonlinear).

2.1.1 The Equations Of Stylized Johansen

We start from the equations as written down in Chapter 3 of Dixon, Parmenter, Powell and Wilcoxen (1992), which we abbreviate to **DPPW**. This contains a description of the Stylized Johansen model and the derivation of these equations.

The equations of the model are shown in Table 2.1.1a. In that table, both the levels and linearized versions of each equation are shown, taken essentially unchanged from DPPW.[1] Notice that, in Table 2.1a, 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 implementation of Stylized Johansen we have chosen a mixed representation, based on the shaded blocks in Table 2.1.1a. 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).

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 given in Table 2.1.1b.

Table 2.1.1b: 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)

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 subscripts 1 and 2 refer to the sectors called s1 and s2, subscripts 3 and 4 refer to the primary factors, labor and capital. Subscript 0 refers to households. (The dollar values in the last three rows of the table have no corresponding DPPW notation.)

---

1 The last 2 rows in Table 2.1.1a, which relate dollar values to prices and quantities, are not explicitly written down in DPPW but, of course, underlie the treatment there. The levels equations are (E3.1.9) [consumer demands], (E3.1.10), (E3.1.12), (E3.1.6), (E3.1.7) and (E3.1.23) [numeraire] in DPPW, while the corresponding linearized equations are (E3.2.1), (E3.2.2), (E3.2.3), (E3.2.4), (E3.2.5) and (E3.2.6) respectively.

Table 2.1.1c: Parameters for Stylized Johansen

Parameters	Meaning	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)

In formulating the equations (see Table 2.1.1d), it is convenient to introduce two sets SECT and FAC. SECT is the set of sectors and FAC is the set of the two factors "labor" and "capital". Note that, since each industry produces a single commodity in this model, the set SECT doubles as the set of commodities and the set of industries (and we can use the terms sector, industry and commodity somewhat interchangeably).

Table 2.1.1d shows the selected equations from Table 2.1.1a, this time using the GEMPACK variables and notation as in Tables 2.1.1b and 2.1.1c. Note that 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. The equations in the TABLO Input file (see section 2.3.1) are taken directly from Table 2.1.1d.

Table 2.1.1d: Equations for Stylized Johansen

```
(E1)   p_XH(i) = p_Y - p_PC(i)                          i in SECT
(E2)   p_XC(i,j) = p_XCOM(j) - [p_PC(i) - p_PC(j)]      i,j in SECT
(E3)   p_XF(f,j) = p_XCOM(j) - [p_PF(f) - p_PC(j)]      f in FAC, j in SECT
(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
(E5)   XCOM(i) = XH(i) + SUM(j,SECT, XC(i,j))           i in SECT
(E6)   XFAC(f) = SUM(j,SECT, XF(f,j))                   f in FAC
(E7)   PC("s1") = 1
(E8)   XC(i,j) = DVCOMIN(i,j) / PC(i)                   i,j in SECT
(E9)   XH(i) = DVHOUS(i) / PC(i)                        i in SECT
(E10)  XF(f,j) = DVFACIN(f,j) / PF(f)                   f in FAC, j in SECT
```

2.2 Data Requirements For A Model

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 for Stylized Johansen, and this is typical of other models, 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.]

2.2.1 The 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 al.l the prices

```
PC(i)                 Price of commodities
PF(f)                 Price of factors
```

we can infer all other levels variables in Table 2.1.1b 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) in Table 2.1.1d 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 the total costs in industry j,

```
SUM(i,SECT,DVCOMIN(i,j)) + SUM(f,FAC,DVFACIN(f,j)).
```

Thus the only data requirements are the dollar values

```
DVHOUS(i), DVCOMIN(i,j) and DVFACIN(f,j).
```

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.

2.3 Constructing The TABLO Input File For A Model

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 in the EQUATIONs;
- the SETs used to describe the different arguments of variables;
- a description of 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).

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 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 understood by example. Hence we launch straight into the preparation of the TABLO Input file for Stylized Johansen.

2.3.1 The TABLO Input File For Stylized Johansen

In this subsection we consider just two equations of Stylized Johansen, namely (E9) and (E4) in section 2.1.1 above. We show how these are written in the TABLO Input file. (We show the full TABLO Input file in Appendix A.)

Consider first the very simple equation (E9) relating prices, quantities and dollar values of household consumption. In the TABLO Input file this equation is written as[2]

```
EQUATION House # Household demand for commodity i #
    (all,i,SECT) XH(i) = DVHOUS(i) / PC(i) ;
```

where

- **EQUATION** is a keyword indicating that what follows is an equation,
- **House** is the name by which this equation is known in the model,
- the words between the hashes **#** form optional additional labelling information which is associated with the equation,
- the quantifier **(all,i,SECT)** indicates that there are really several equations, one for each sector, and
- the semicolon ; marks the end of this part of the 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 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 3.1, for example) and are available for use in linearized equations in the TABLO Input file without further explicit declaration. (See, for example, the EQUATION named "Price_formation" discussed later in this section.)

---

2 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 fact that SECT is a set containing two sectors "s1" and "s2", 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.[3] 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[4]

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

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

---

3 This part of the implementation of a model via GEMPACK is somewhat analagous to the so-called calibration phase carried out with other software.

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

5 The actual name of this file on the 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.

```
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, to check the values of say ALPHAFAC, one of the following statements could be added:

```
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.

The complete TABLO Input file is shown in Appendix A. It includes all the statements above (except the DISPLAY and WRITE statements). Appendix A also contains commentary about features of the TABLO Input file not mentioned above.

2.4 Linearized Representations and Update Statements

For some time (that is, prior to Release 5.0 in 1993), GEMPACK only allowed linearized equations in TABLO Input files. In linearized representations, the linear variables (the changes or percentage changes) are usually declared explicitly and separately from the levels variables. In addition, all levels equations must have been linearized by hand by the modeller to equations which are linear with respect to the linear variables. For example, the linearized representation of the levels equation D = P*Q is

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

Often linearized equations involve both linear variables and levels variables. For example, the linearized version of the levels equation

$$X = Y + Z$$

would often be written as

$$X*p\_X = Y*p\_Y + Z*p\_Z.$$

If this linearized equation were written in a linearized TABLO Input file, the percentage changes p_X,p_Y,p_Z would be declared as linear variables and the levels variables would be declared as COEFFICIENTs, as shown in Figure 2.4.

Figure 2.4: TABLO Statements for one Equation

```
VARIABLE (LINEAR) p_X ;
VARIABLE (LINEAR) p_Y ;
VARIABLE (LINEAR) p_Z ;

COEFFICIENT X ; Y ; Z ;
UPDATE X=p_X ; Y=p_Y ; Z = p_Z ;

EQUATION (LINEAR) eq1  X*p_X = Y*p_Y + Z*p_Z ;
```

In such a case, the software must be told explicitly the connection between the linear variables and their associated levels variables (COEFFICIENTs). This is done via so-called UPDATE statements. For example, the statement

```
UPDATE X = p_X ;
```

in Figure 2.4 indicates that p_X denotes the percentage change in the levels variable X.[6] Similarly,

```
UPDATE (CHANGE) W = c_W ;
```

would indicate that the linear variable c_W represents the change in levels variable (COEFFICIENT) W.

In linearized TABLO Input files (see, for example, ORANI-F in Horridge *et al.* (1993)),

- dollar values are read in but levels of prices and quantities do not usually need to be considered explicitly,
- percentage changes in prices and quantities are explicit linear variables but percentage changes in the associated dollar values are usually not included.

---

6 When linear variables are declared explicitly, there is no restriction that their names must begin with "p_" or "c_" or have the same stem as the levels variable. This is why "update" statements connecting the linear variables and their associated levels version are required.

In such files, the dollar values read must be updated via the associated linear price and quantity variables. Most update statements are of the form

```
UPDATE  DV = p * q;
```

where DV is a COEFFICIENT holding a dollar value and p and q are linear variables denoting percentage changes in the relevant price and quantity.[7]

2.5 Different Representations (Levels, Linearized or Mixed)

As indicated above, an economic model can be specified by giving all levels equations, all linearized equations or a mixture of linearized and levels equations. In the Stylized Johansen TABLO Input file we used a mixed representation.

For discussions of the merits of working with different representations of models, see Harrison *et al.* (1993) and Hertel *et al.* (1992). Since they all produce the same results, our main advice is to work with whichever representation seems most natural or convenient.

Here, to help reinforce the difference between possible representations, we take the simple (non-economic) example with just one equation $Y=X^3$, and give both a levels and a linearized TABLO Input file for this (see Figures 2.5a and 2.5b). (Of course, a mixed one is not possible since this is just a single

Figure 2.5.a: Levels TABLO Input file

```
! Levels version of Y=X-cubed !
VARIABLE (LEVELS,CHANGE) Y ;
VARIABLE (LEVELS,CHANGE) X ;
READ X FROM TERMINAL ;
FORMULA & EQUATION eq1 Y = X^3 ;
```

Figure 2.5b: Linearized TABLO Input file

```
! Linearized version of Y=X-cubed !
VARIABLE (LINEAR,CHANGE) dX # change in Y # ;
VARIABLE (LINEAR,CHANGE) dX # change in X # ;
COEFFICIENT X # Levels value of X # ;
UPDATE (CHANGE) X = dX ;
READ X FROM TERMINAL ;
EQUATION (LINEAR) eq1 dY = 3*X^2*dX ;
```

---

7 This UPDATE statement gives rise to the formula

new_DV = old_DV[1 + (p+q)/100]

for calculating the new DV from the old DV and the percentage changes p and q. (This is done after each step of a multi-step calculation - see section 4.2.)

equation.) In these files, "eq1" is the name by which the equation is known in the TABLO Input file. We have chosen to use change variables in both cases since it is quite possible for X and Y to be positive, zero or negative. The initial solution is treated differently in the two cases.

- In the levels case, X is read from the terminal and the initial value of Y is given by the FORMULA (which is also the EQUATION).
- In the linearized case, the initial value of X is read. Although X appears explicitly in the linearized equation in Figure 2.5b, Y does not, and so we do not need to declare a COEFFICIENT Y in this case (or give it initial values).

2.5.1 TABLO Linearizes Levels Equations

When the program TABLO processes a TABLO Input file, it automatically linearizes any levels equations; indeed TABLO converts the whole TABLO Input file to a linearized TABLO Input file (which is called the **associated linearized TABLO Input file**). After this, all interaction with the software about the model proceeds as if this associated linearized TABLO Input file were the actual TABLO Input file.[8] For example, if we begin from the levels TABLO Input file in Figure 2.5a above, when we specify the closure, we must refer to the linear variables c_X and c_Y rather than the levels ones.

When producing the associated linearized file, TABLO inserts UPDATE statements as required to connect the linear and levels variables. For example, when the file in Figure 2.5a above is processed, the statement

```
UPDATE(CHANGE) X = c_X ;
```

would be included in the associated linearized TABLO input file.

2.6 Example Models

The following models are usually supplied with GEMPACK:

Stylized Johansen (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),

GTAP, the Global Trade Analysis Project's model for analysing trade issues, as documented in Hertel and Tsigas (1993),

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

---

8 This is largely because, in the early versions of GEMPACK, only linearized TABLO Input files were accepted.

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).

Other models implemented and solved via GEMPACK include

- single-country models of the Philippines (Warr *et al.* (1993) and Borrell *et al.* (1994)), Indonesia (Trewin *et al.* (1993)), Zimbabwe (Quirke *et al.* (1993)), Sri Lanka (Centre for International Economics (1992)), China (Gao (1993), Huang (1993) and Martin (1991)), Papua New Guinea (National Centre for Development Studies (1990) and Woldekidan (1993)),
- several extensions of ORANI including FH-ORANI (Dee (1989)), MONASH (Adams *et al.* (1993)), and a fully intertemporal version (Malakellis (1992)),
- multi-country models such as SALTER (Jomini *et al.* (1991)), Asian models (Hughes (1990), Mai (1993), Suphachalasai (1989) and Yang (1994)) and an intertemporal model of the global meat industry (Harris *et al.* (1992)), and
- an intertemporal model of a Ramsey Problem (McDougall (1994)).

GEMPACK has also been used for database manipulation as in the FIT facility (James *et al.* (1993)).

GEMPACK software makes it easy to transfer models between different computers (including different operating systems). For example, the main theory of the model is all in the TABLO Input file which is an ASCII text file readily transferred to other computers. Hence it is easy to obtain models from other modellers using GEMPACK (see section 8 below).

3 CARRYING OUT SIMULATIONS

In this section, we describe how simulations are carried out in GEMPACK and how simulation results are reported and interpreted. We illustrate these general points by considering, in some detail, a simulation with Stylized Johansen.

3.1 Interpreting The Results Of A Simulation

When a simulation is carried out, the software typically reports changes or percentage changes in selected variables and produces updates (that is, post-simulation) data. The initial (that is, pre-simulation) data is also important in interpreting results. We illustrate these points by taking an example simulation with Stylized Johansen.

3.1.1 A Simulation With Stylized Johansen

The most commonly-used closure of Stylized Johansen is the one in which supplies of the two factors, labor and capital, are taken as exogenous, and all the remaining variables are endogenous. Here we have chosen to look at the

simulation in which the supply of labor is increased by 10 per cent and the supply of capital is held fixed.

We have taken the initial data to be as in Table E3.3.1 of DPPW; we reproduce this here as Table 3.1.1a. 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.

Table 3.1.1a: Input-output Data Base for Stylized Johansen

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

Some of the results of this simulation are given in Table 3.1.1b. (This shows the values of certain of the endogenous variables essentially in the form output by the GEMPACK program GEMPIE.)

Table 3.1.1b: Part of Simulation Results File

```
PAGE 1  Labor Supply Increase
p_Y Total nominal household expenditure
    5.8853

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

p_XH    (SECT) Household demand for commodity i
    s1                  s2
    5.88536.8993

p_XF    (FAC,SECT) Factor inputs to industry j
  p_XF(-,s1) results where '-' is in set 'FAC'.
    labor               capital
    10.0000             0.0000
  p_XF(-,s2) results where '-' is in set 'FAC'.
    labor               capital
    10.0000             0.0000

p_DVHOUS (SECT) Dollar value of household use of commodity i
    s1                  s2
    5.88535.8853
```

The results in Table 3.1.1b mean that, if the supply of labor is increased by 10 per cent and the supply of capital is held fixed, then, for example,

(1) households will consume 6.8993 per cent more of commodity 2 than they did previously (the 'p_XH' result for commodity 2),
(2) the price of commodity 2 will fall by 0.9486 per cent (the 'p_PC' result for commodity 2), and
(3) the dollar value of household consumption of commodity 2 will rise by 5.8853 per cent (the p_DVHOUS("s2") result).

Recall that, in the TABLO Input file for Stylized Johansen (see section 2.3.1), initial levels values of prices and quantities are calculated by setting prices to 1, which 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) household consumption of commodity 2 has increased to 4.2760 million units (6.8993 per cent more than the original 4 million units),
(2) the price 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
(3) the dollar value of household consumption 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).

Of course the updated values in (1), (2) and (3) above should be related since dollar value should equal price times quantity. Note that this is true since, from (1) and (2) 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 (3). This confirms that the solution shown in the results file satisfies the levels equation connecting price, quantity and dollar value of household consumption of this commodity.

From the results of the simulation, it is easy to infer the new levels values of all quantities of interest in the model (prices, quantities and dollar values). Indeed, the updated data file produced during the simulation contains the new levels values for the quantities read in initially from the data base.

3.2 Specifying A Simulation

In order to specify the details for carrying out a simulation, it is necessary to give details of

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

(All of this information is shown schematically in Figure 3.2.)

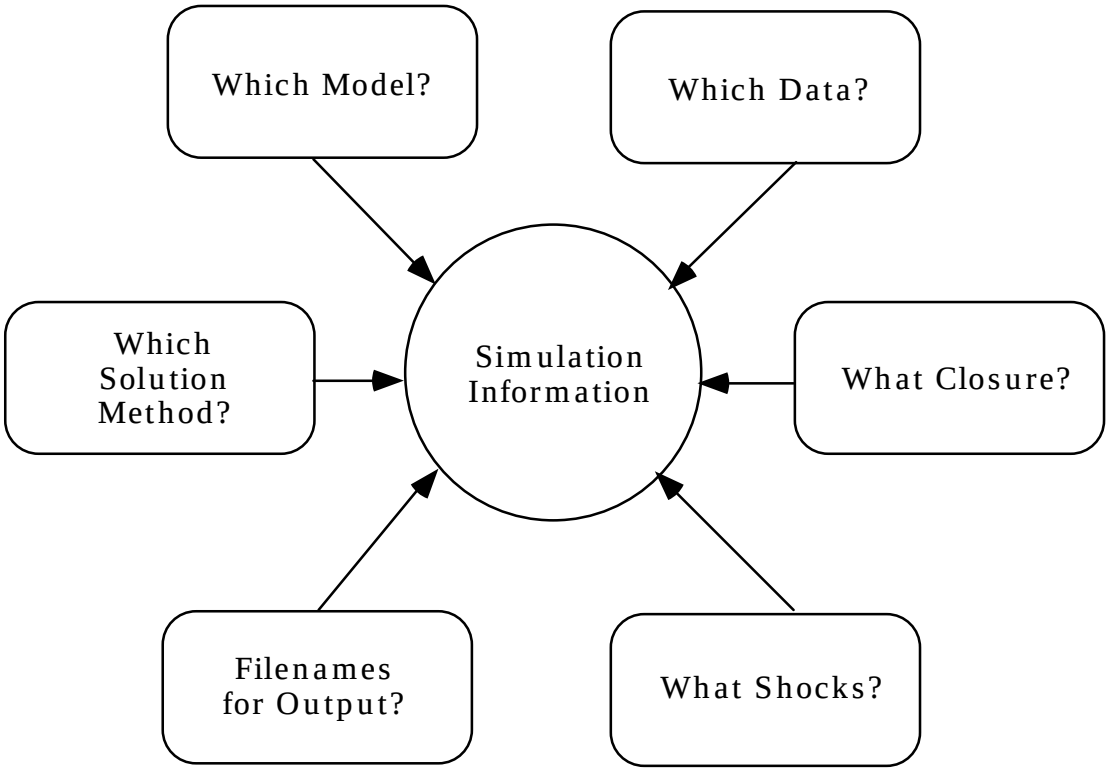

Figure 3.2: The Information Required to Specify a Simulation

GEMPACK uses small text files called Command files to specify a simulation. The Command file used to specify the simulation described in section 3.1.1 above is shown in full in Figure 3.2.1; we describe some of its features in section 3.2.1 below. The syntax of Command files has been chosen in the hope of providing an easily understood and self-contained record of the simulation.[9]

3.2.1 An Example Command File

The statement

```
auxiliary files = sj ;
```

in the Command file shown in Figure 3.2.1 tells the program carrying out the simulation which model to work with, since these auxiliary files are just a processed version of the TABLO Input file for the Stylized Johansen model (see section 3.3 below). The statement

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

tells the program to read base data from the file SJ.DAT (which contains the data in Table 3.1.1a above). The statements

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

---

9 We are grateful to Peter Wilcoxen for suggesting the use of Command files and for providing us with a prototype implementation.

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 ;
```

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

Figure 3.2.1: Example of a GEMPACK Command File

```
!
! GEMPACK Command file which 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 ;
! 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.)
1,2,4-step solutions plus extrapolation.  ;
! Solution method information
method = euler ;
steps = 1 2 4 ;
! Equations file information
equation file = sj ;
model = sj ;
version = 1 ;
identifier = Stylized Johansen.  Standard data.  ;
! Options
extrapolation accuracy file = yes ;
! End of Command file
```

The statement

```
solution file = sjlb ;
```

specifies the name of the Solution file to contain the solution of the simulation. The statement

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

names the file to contain the updated (that is, post-simulation) data. (The name includes 'LB' to remind us that this data depends on the labor shock.) The **verbal description** of the simulation, which can be several lines of text, goes on the Solution file and is transferred to the results file. This can be used to describe the salient features of the simulation. The statement

```
verbal description = ..... ;
```

gives 4 lines of text for the verbal description in this case. (The ';' indicates the end of this description. Note that all statements in GEMPACK Command files must end with a semicolon';'. Lines beginning with an exclamation mark ! are comments.)

With GEMPACK, there are 4 related solution methods one of which can be chosen for a simulation. These are introduced in section 4.3 below. The statements

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

tell the program to use Euler's method based on 3 separate solutions using 1, 2 and 4 steps respectively. The accuracy of the solution depends on the solution method and the numbers of steps. The statement

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

asks the program to produce a so-called **Extrapolation Accuracy file** which provides information about the accuracy of the solution (see section 4.3 for more details).

The program carrying out the simulation usually produces a so-called **Equations file** (see section 4.1 below) which contains the numerical linearized equations of the model. (This can be used as a starting point for calculating Johansen solutions - see section 4.4 below.) The statements

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

specify the name of the Equations file, the model name, the version number and a model identifier.

3.3 Steps In Carrying Out A Simulation

The program GEMSIM is a general-purpose program for carrying out simulations with different models. It can be used to carry out simulations with any model implemented in GEMPACK.

For small or medium-sized models, GEMSIM runs quickly and there is no need to use any alternative. However, for large models (for example, the ORANI model of the Australian economy, which has over 100 sectors), there is an alternative way of proceeding which can result in much quicker simulations. This involves asking TABLO to write a special-purpose Fortran program (called a **TABLO-generated program**) to capture the theory of the model (rather than to produce the computer files used by GEMSIM). A TABLO-generated program is not general-purpose, but specific to one model.

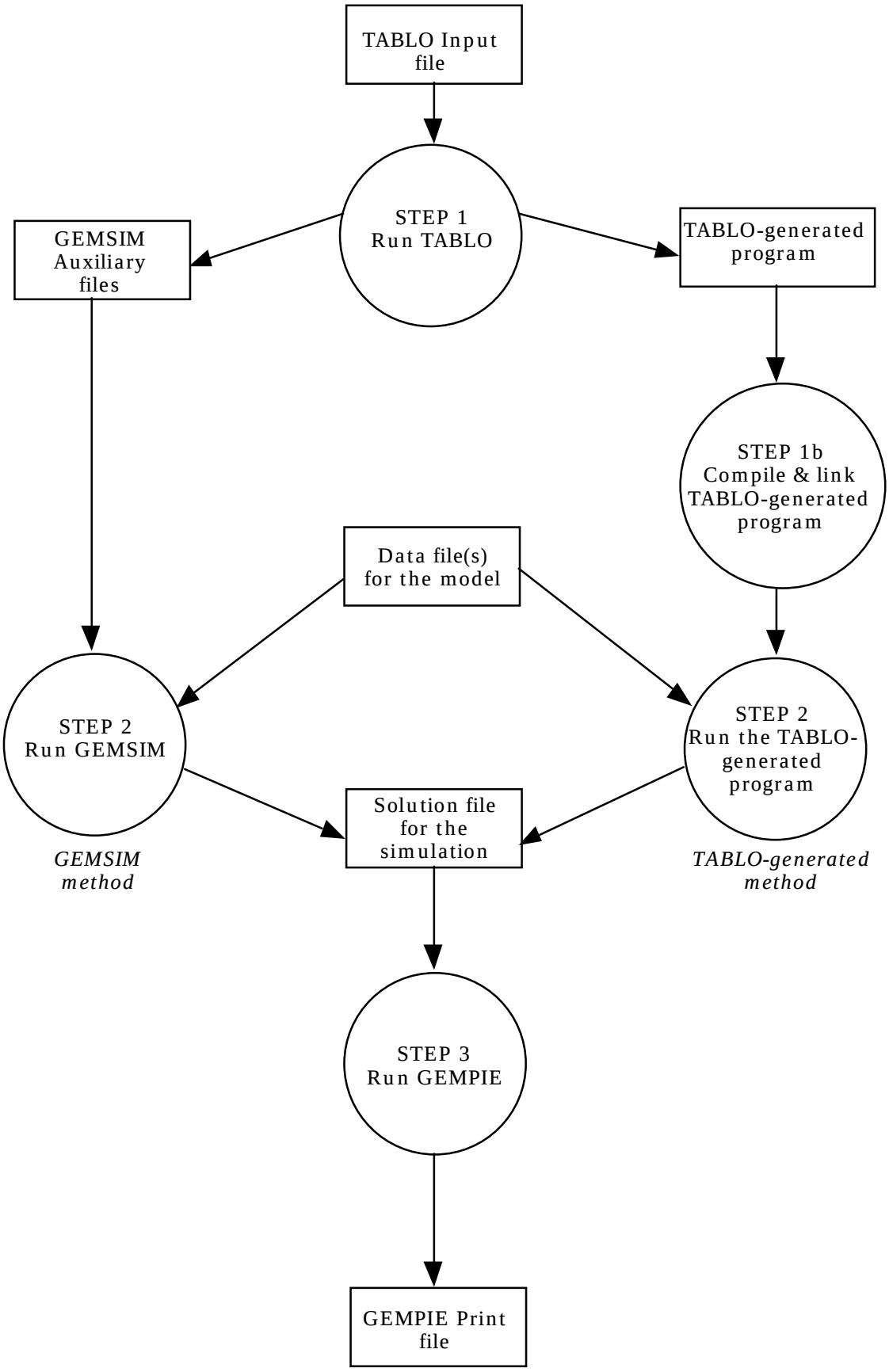

Figure 3.3: Steps in carrying out a Simulation

The GEMSIM method of carrying out simulations is illustrated on the left hand side of Figure 3.3 while the TABLO-generated method is on the right hand side.[10] The three steps are:

Step 1. Computer Implementation of the Model

Process the TABLO Input file for the model by running the GEMPACK program TABLO. The user can choose either the GEMSIM method or TABLO-generated program method. If the TABLO-generated program route is chosen, after TABLO has finished, the TABLO-generated program is compiled using an appropriate Fortran compiler, and linked to the GEMPACK library of subroutines.

Step 2. Simulation

Run the GEMPACK program GEMSIM or the TABLO-generated program. Specify which base data are to be read and describe the closure (that is, the exogenous and endogenous variables) and the shocks as described in section 3.2. The program then computes the solution to the simulation and writes the results to a so-called Solution file. It also produces updated data.

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).

Often many different simulations are carried out on the same model, for example, with different closures and/or shocks, or starting from different base data. In these cases, Steps 2 and 3 are repeated but not Step 1. However Step 1 must be repeated if the TABLO Input file for the model is changed in any way.

The input to specify a simulation is essentially the same for both GEMSIM and the TABLO-generated program. Command files, as in section 3.2, can be used for both. These programs can also be run interactively by responding to prompts from the program.

3.4 Different Closures And Shocks

Most general equilibrium models have several different closures; which one to use depends on the purpose of the simulation in question. In section 3.4.1 below, we show how different closures can be specified on a GEMPACK Command file.

Especially when the model is used for forecasting, a large number of shocks may need to be specified. In section 3.4.2 below, we show how these can be specified on a GEMPACK Command file.

---

10 The TABLO-generated method requires a suitable Fortran compiler, and also a source-code (rather than executable image) version of GEMPACK. Accordingly, this method is not available with the Demonstration or Executable Image versions of GEMPACK (see section 11).

3.4.1 Different Closures

With the Stylized Johansen model, the usual closure has supplies of the 2 primary factors (labor and capital) exogenous and all other variables endogenous. Alternative closures are to have the supply of one factor and the price of the other factor exogenous and all other variables endogenous. The statements in a Command file for the first of these closures can be

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

(as shown in Figure 3.2.1) while the statements

```
exogenous p_XFAC("labor") p_PF("capital") ;
rest endogenous ;
```

specify the closure in which the supply of labor and the price of capital are exogenous.

With large general-purpose models such as ORANI-F (see Horridge *et al.* (1993)) or GTAP (see Hertel and Tsigas (1993)), there are usually a large number of different closures. For example, in ORANI-F (see section 5 of Horridge *et al.* (1993)),

(a) the numeraire can be either the exchange rate 'phi' or the domestic CPI 'p3tot';
(b) it may be appropriate to take aggregate employment 'employ_i' exogenous and the real wage rate endogenous, or vice versa;
(c) it may be appropriate to exogenise household consumption (via the variable 'w3lux') or to exogenise the balance of trade (via variable 'delB').

The usual (general equilibrium) closure of GTAP has supplies of land, labor and capital exogenous (in all regions) and supplies of all other commodities and all commodity prices endogenous. It is also useful to consider partial equilibrium closures to illustrate differences between policies and/or to analyse different policies. In one such closure, a multi-region general-equilibrium (MRGE) closure in the GTAP literature focusing on food (see Hertel and Tsigas (1993)), supplies of all commodities except food in all regions are exogenous and all commodity prices except for those of food and land in all regions are exogenous; with this closure some equations which usually hold in a general equilibrium model are effectively turned off to give a partial equilibrium model. Usually the variable 'walraslack' is endogenous and its value is used to check that Walras law holds; in the MRGE closure mentioned above, this variable is set exogenous (and not shocked) in order to ensure that Walras law still holds in this partial-equilibrium version of the model.

On a GEMPACK Command file, the usual way of specifying a closure is to list the exogenous variables and to conclude with the statement “rest endogenous ;".

Once one standard closure has been set up and saved (closures are saved on so-called Environment files), it may be easier, and more informative, to specify an alternative closure by saying how it differs from the standard one. For example, if a standard closure for ORANI-F has been saved on Environment file ORF and we wish to specify a different closure in which the domestic CPI 'p3tot' and the balance of trade 'delB' are exogenous (rather than variables 'phi' and ‘w3lux'), the Command file statements could be

```
modify closure on Environment file orf ;
swap p3tot = phi ;
swap delB = w3lux ;
```

using "swap" statements or, alternatively,

```
modify closure on Environment file orf ;
exogenous p3tot delB ;
endogenous phi w3lux ;
```

in which the new status of the relevant variables is indicated explicitly.

3.4.2 Shocks

The values of shocks can appear directly on a Command file or on a text file whose name is given on the Command file. For example, for the ORANI-F simulation discussed in section 7 of Horridge *et al.*(1993), the Command file may contain statements

```
shock p3tot = 34.01 ;
shock delx6 = file delx6.shk ;
shock pf0cif = uniform 23.64 ;
```

The first line gives the shock to the domestic CPI 'p3tot'; it says that this should be increased by 34.01 per cent. The second line says that the values of the shocks to the 13 different components of variable 'delx6' (changes in stocks in the 13 different sectors) can be read from the text file 'delx6.shk'. The third line says that the same shock (this is what the word 'uniform' means), namely an increase of 23.64 per cent, should be given to each of the 13 components of variable 'pf0cif'; these shocks are part of the changes in the terms of trade expected over the period covered by the forecast.

In some cases, values of shocks may be calculated most easily via a TABLO Input file constructed explicitly for this purpose. For example, with GTAP, it is convenient to construct a TABLO Input file which reads the existing data, calculates current distortions and then calculates changes required to remove these distortions; these values can be written to text files (using WRITE statements). These text files can serve as the shock files for simulations with the model intended to give information about changes once some or all of the distortions are removed (for example, after various GATT policy changes are implemented by some or all countries).

4 HOW GEMPACK SOLVES THE EQUATIONS

We first describe how approximate solutions, know as Johansen solutions, are calculated. This leads on to accurate multi-step solutions, and then to the different solution methods available within GEMPACK. Finally, in section 4.4, we describe how the GEMPACK program SAGEM can calculate several Johansen solutions simultaneously.

4.1 Johansen Solutions

Johansen solutions are calculated by solving the linearized equations of the model once (or in just one step) 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

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

where

C is the m x p matrix of coefficients of the equations,
$\mathbf{z}$ is the p x 1 vector of all the variables of the model,
m is the total number of equations,
p is the total number of variables.

In general, m is less than p in the system of equations in (1) above, so in a simulation (Johansen or multi-step),

(p -m) of the variables are exogenous,
the remaining m variables are endogenous, and
shocks (usually percentage changes) are given to some of the exogenous variables.

For example, for Stylized Johansen, the total number of variables p is 29 and the total number of equations m is 27, so we need 2 exogenous variables. We can shock either 1 or 2 of these exogenous variables.

Once the exogenous/endogenous split has been chosen, the system of equations $C\mathbf{z} = \mathbf{0}$, as in (1) above, becomes

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

where $\mathbf{z_1}$ and $\mathbf{z_2}$ are respectively the (column) vectors of endogenous and exogenous variables, A is m x m and D is m x (p-m). 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 $\mathbf{z_2}$. Once these are known, we have a system

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

to solve (where $\mathbf{b}$ is an m x 1 vector). It is the solution $\mathbf{z_1}$ of this matrix equation (3) which is the Johansen solution[11] of the simulation.[12]

Because the levels equations of the model are usually nonlinear, the results of this calculation are only approximations (sometimes good ones and sometimes not-so-good ones) to the corresponding solution of the levels equations of the model. Accurate solutions require multi-step calculations, which we now describe.

4.2 Multi-Step Solutions

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.

---

11 This name pays tribute to Johansen who pioneered this way of obtaining useful approximate solutions of general equilibrium solutions around 1960.

12 The matrix A in equation (3) is usually sparse in the sense that most of its entries are zero. GEMPACK uses the Harwell Laboratory's sparse matrix routines MA28 (see section 9.7) 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.

Figure 4.2 below makes this easy to visualize. 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 .

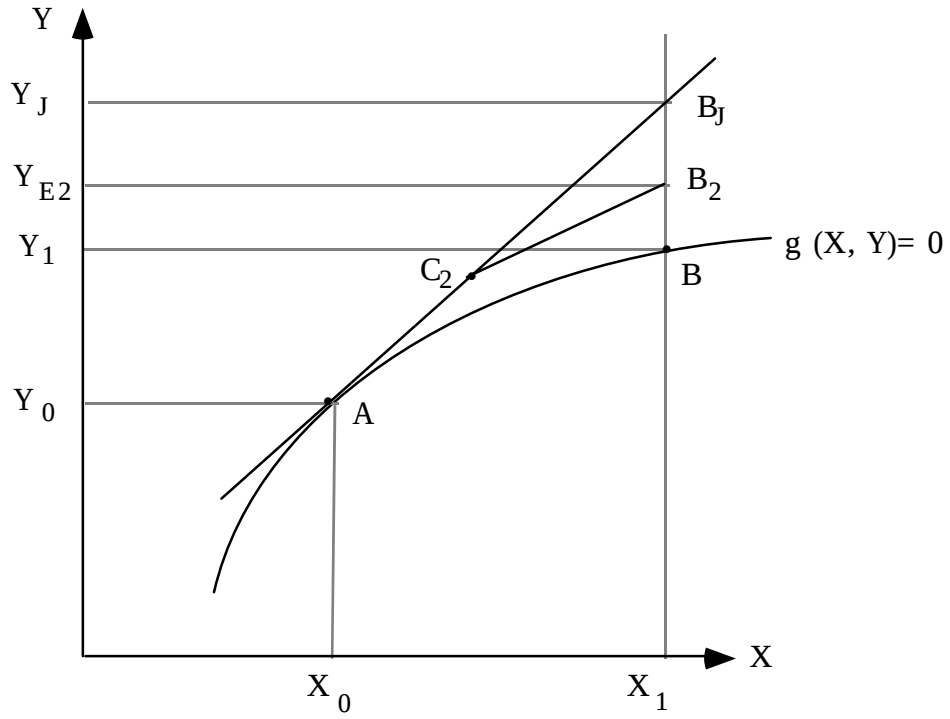

Figure 4.2: Multi-step solution using Euler's method

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 4.2), 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.

In general, the more steps the shocks are broken into, the more accurate will be the results.

4.3 Solution Methods And Extrapolation

One way of increasing accuracy of solution is to increase the number of steps in a multi-step solution. 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**.

GEMPACK can solve the equations using one of the four related solution methods: Johansen, Euler, Gragg or the midpoint method. Gragg's method is often 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. (The midpoint method is similar to Gragg's method.)

To illustrate these points, we show below the different results for the percentage change in household expenditure 'p_Y' in the Stylized Johansen model for the simulation in section 3.1 above, in which labor supply is increased by 10 per cent and capital remains infixed supply. Table 4.3 shows Euler and Gragg results for different step numbers and extrapolations based on them. Note that the exact result is 5.88528.

Table 4.3: Multistep and Extrapolated Results

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[13]	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.)

An **Extrapolation Accuracy file** can be produced to show how accurate the solution is for each endogenous variable. The separate columns show the results

---

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

for the different multi-step solutions calculated, and the last column of results is the extrapolated result. When 3 different multi-step results are used for extrapolation (which is what we recommend), the last two columns give conservative information about the number of figures of accuracy of each result.

4.3.1 Connection With Initial Value Problems

The kind of simulations GEMPACK is designed to solve can be converted to a class of well-known problems called Initial Value problems. Details of this conversion are given in Appendix B.

There are many different methods for solving Initial Value problems, as can be seen by consulting almost any numerical analysis textbook, for example, Chapter 6 of Atkinson (1989). GEMPACK makes available three of the simplest and best-known methods, namely Euler's method, the midpoint method and Gragg's method (also known as the modified midpoint method); see, for example, Chapter 6 of Atkinson (1989) or Chapter 15 of Press *et al.* (1986) for a description of these methods. All of these methods solve an Initial Value problem by approximating the solution curve by a sequence of straight-line segments, as in Figure 4.2 above.

Different Initial Value solution methods have different orders of errors. Euler is an order 1 method in the sense that, if the number of steps is multiplied by N, then the errors are approximately divided by N, while Gragg and midpoint are order 2 methods which means that the errors are approximately divided by N^2 .

Within GEMPACK, we recommend extrapolating from three different solutions (each calculated with a different number of steps - for example, 2,4,6-step solutions); this is usually the most efficient way of obtaining accurate solutions of Initial Value problems. When extrapolating from three solutions using Euler's method, if the number of steps are all multiplied by N, then the errors in the extrapolated solution will be divided approximately by N^3 . (For example, the errors in the result extrapolated from 6,12,18-step Euler solution is expected to be twenty-seven times smaller than that from a 2,4,6-step Euler solution; here N=3.) For Gragg and the midpoint method, the errors in the extrapolated result are expected to be divided by N^6 (instead of N^3 for Euler) . See, for example, Atkinson (1989) or Press *et al.* (1986) for details about general extrapolation errors; see Pearson (1991) for a discussion with reference to GEMPACK solution methods.

4.4 Several Johansen Simulations At Once

The GEMPACK program SAGEM can be used to compute several Johansen solutions at once. Although Johansen solutions are less accurate than multi-step ones, carrying out Johansen simulations 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 more quickly than 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 GEMSIM or the TABLO-generated program (as in Step 2 of Figure 3.3). This Equations file is just a binary file containing the entries of the matrix C in equation (1) of section 4.1. The closure and shocks are input to the program SAGEM, which solves the set of equations (once), and computes a solution for each of the shocks.

The results of a SAGEM run which carries out two Johansen simulations with the Stylized Johansen model are shown in Table 4.4. They 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 multi-step solution produced in section 3.1 above. However, the advantage of Johansen results is that they can be scaled and combined to estimate the cumulative effect of any set of shocks. 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 percent increase (column 1 in Table 4.4) by 10. The results can be compared with those of the multi-step simulation in Table 3.1.1b 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.)

Table 4.4: Individual Column Results for Johansen simulations

PAGE 1 All shocks 1

		p_XFAC 1	p_XFAC 2	TOTALS
		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_XH	(SECT) Household demand for commodity i			
1	s1	0.60000	0.40000	1.00000
2	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 ind 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.			

The main reason for using SAGEM rather than running GEMSIM or the TABLO-generated program is that SAGEM can produce the Johansen results of several simulations more quickly (that is, using less computing resources) than one single, more accurate, multi-step simulation result. The CPU time required for SAGEM to produce several individual column results is only about the same as that required for one step of a multi-step calculation since the cost of the LU decomposition outweighs the cost of solving for several columns.

5 CONDENSING LARGE MODELS

In many cases models need to be reduced in size before it is practical to solve the linearized equations. For example,

- with its usual disaggregation of about 110 sectors, the ORANI model of the Australian economy (see Dixon *et al.* (1982)) has over a million equations and several million unknowns (that is, in the notation of section 4.1, m is about one million and p is several million). This could not be solved without condensation even on very large mainframes.
- the 10-commodity 7-region version of GTAP, the Global Trade Analysis Project's model (see Hertel and Tsigas (1993)), has about 22000 equations and 32000 unknowns. While this could be solved directly on large mainframes, it would not be solvable on PCs with modest memory (say 8Mb) in this form. However, with TABLO's condensation facility, it can be reduced to around 7000 equations (or fewer, if necessary) and solved on such a PC.

The point of condensation is to reduce the size of the system of equations (that is the number of equations and the number of variables) that must be solved directly. The main ideas behind condensation, namely substitution and omission of variables, are easy to understand; they are explained below. In the early days of applied general equilibrium modelling, these were carried out with pencil and paper for models such as ORANI; such calculations were very time consuming and somewhat error prone. Now TABLO can be asked to do these substitutions and omissions; it does them quickly and reliably. All that the user has to do is to tell the program which variable to substitute out and the name of the equation to use to substitute it out. Condensation is so routine that now modellers change their condensation for different groups of simulations.

Below we describe briefly the different ways of condensing a model, namely substituting out variables (or backsolving for them, which is much the same as substitution) and omitting variables. Variables substituted out (or backsolved for) must be endogenous while those omitted must be exogenous and not shocked in the current group of simulations.

5.1 Substituting Out Variables

Suppose that the linear variable **x** (all components of it) is to be substituted out 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,c,COM) B5(c)*(x(c) + y(c)) = 0
```

becomes

```
(ALL,c,COM) B5(c)*([A6(c)*y(c)+z(c)] + y(c)) = 0.
```

An equation nominated 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 warns of this potential problem during the condensation stage. If the user proceeds with the substitution and some value of A8 is indeed zero, the error will be detected when GEMSIM or the TABLO-generated program runs the simulation.

Substituting out a variable with k components reduces by k the number of rows and the number of columns of the matrix in the system of equations to be solved (that is, reduces both m and p in section 4.1 by k).

5.2 Backsolving for Variables

When a linear variable is substituted out, it is eliminated from all equations in the condensed system and its values are not calculated (and so cannot be reported) in the solution of a simulation.

In principle, the values of a variable substituted out could be calculated after each step of a multi-step simulation by substituting the values of variables in the condensed system into the expression used to substitute out the variable in question. For example, if variable **x** has been substituted out using the equation

```
(ALL,i,COM) x(i) = A6(i)*y(i) + z(i)
```

and if variables **y** and **z** remain in the condensed system, after each step of a multi-step simulation, we could calculate the values of x(i) by substituting in the known values of A6(i), y(i) and z(i) into the right-hand side of the equation above. This is known as **backsolving** for variable **x**.

Instead of substituting out a variable, it can be marked as one that we may want to backsolve for, to obtain its simulation values. The variable and equation in question are still eliminated from the condensed system. However, later in a simulation using GEMSIM or the relevant TABLO-generated program, the variable can be chosen as one of the variables to be included on the Solution file and its values will be calculated by backsolving.

5.3 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 a variable with *k* components is omitted, this reduces the number of columns in the matrix C by *k* (but does not change the number of rows).

If, in another group of simulations, these omitted variables are to be shocked (or made endogenous), a different condensation can be carried out in which these are not omitted (but perhaps others are).

5.4 The Effect Of Substitutions On Computational Complexity

Here we discuss briefly factors affecting the computational complexity of the calculations required to set up (but not solve) the system Cz=0 (see equation (1) in section 4.1 above) of linear equations for the condensed system. The computational complexity is a measure of the amount of arithmetic required to calculate the entries of the matrix C; the processing (CPU) time required is usually proportional to the complexity.

In general, the computational complexity increases when a substitution is made. To see intuitively why, consider the example of substitution given above in section 5.1. The simple expression x(i) is replaced in possibly many places by the more complicated expression A6(i)*y(i)+z(i).

To reduce this increased complexity where possible, TABLO carries out some analysis of expressions during condensation. For example, as we illustrate in the example below, TABLO may automatically define a new coefficient to stand for a complicated term. This avoids having to recalculate this term several times.

Example

Suppose, for example, that a variable x(i,j) with two arguments is being substituted out, and suppose that the equation being used to substitute it out has another term A(i,j)*y(i), where A(i,j) is a COEFFICIENT and y(i) is a linear VARIABLE. When TABLO is making this substitution, it replaces all occurrences of variable 'x' in all other equations. Suppose that another equation has a term

```
SUM( j, IND, B(i,j)*x(i,j) )
```

in it. When the substitution is made for x(i,j), this equation will contain a term

```
SUM( j, IND, B(i,j)*A(i,j)*y(i) )
```

which can be rewritten as

```
[SUM( j, IND, B(i,j)*A(i,j)] * y(i)
```

where the order of the SUM and product (*) have been changed. Here, if this equation is later used to make a substitution, this complicated term (the sum of the products B(i,j)*A(i,j)) may enter several other equations and have to be calculated several times. Since this calculation must be done at least once, and to forestall it being done

several times, TABLO will choose to introduce a new coefficient say C00234(i) and a formula setting

```
(all,i,COM) C00234(i) = SUM(j,IND, B(i,j)*A(i,j) )
```

However, this efficiency gain requires extra memory (namely that required to store the values of C00234(i,j) for all relevant values of i and j).

We found that, with a version of the ORANI model of Australia, this automatic introduction of new coefficients and formulas reduced the complexity (and the CPU time) of the calculations setting up the equations Cz=0 by over 60 per cent.

6 DATA PREPARATION AND RESULT REPORTING

GEMPACK includes some facilities for data preparation and result reporting. However the design plan for GEMPACK recognises that there are excellent tools outside of GEMPACK for these tasks; accordingly GEMPACK aims at making it easy for users to move data into and out of GEMPACK and to move results to other software. The interfaces between GEMPACK and other software are text files, including comma-separated-value (CSV) text files to/from spreadsheet programs. We give some details in this section.

6.1 Data Preparation

As explained in section 9.5 below, data for large models is typically stored in GEMPACK as binary files (though text data files are also allowed in GEMPACK). GEMPACK has developed binary files called Header Array. Each **Header Array** file can contain many different arrays of data; each array is identified by its own 4-character "header". For example, the statement

```
READ SALES FROM FILE iodata HEADER "ABCD" ;
```

indicates that the data associated with COEFFICIENT SALES is to be found at the header 'ABCD'. The internal structure of these files has been designed so that the software can move quickly to this part of even a large file.

Because these Header Array files are special to GEMPACK, software for creating them, for editing (that is, modifying) the data on them, and for examining the data on them is part of GEMPACK. The program MODHAR can be used to create such a file or to modify it. In either mode, it can accept data from text files, including CSV files exported from spreadsheets, or from other Header Array files, or entered interactively. Typically, for a new model, the different arrays of data are prepared outside GEMPACK, and then MODHAR is run to put them together into one or more Header Array files. Subsequent changes (for example, changing the values of some of the behavioural parameters) can be done using MODHAR or can be carried out by going back to the original sources (for example, spreadsheets) and making the modifications there.

Some data preparation tasks can be carried out using TABLO itself. It is often easy to write TABLO Input files which just do data manipulation (for example, to aggregate data). The program SEEHAR for examining the data on a Header Array file can output the data in a form suitable for printing, or can produce output in CSV form so that it can be easily imported into a spreadsheet for manipulation there.

6.2 Result Processing And Reporting

The GEMPACK program SLTOHT can be used to convert simulation results to various kinds of text files (including CSV) or to Header Array files. Often the results of two or more simulations must be combined or compared during report writing. Many users find that they can do this best by first importing the results into spreadsheet programs for manipulation and/or preparation of graphical reports, and then moving these into a word processor.

When particularly number-intensive or complicated post-simulation manipulations are required, some users convert the results to a Header Array file and then write a data-manipulation TABLO Input file to read the solutions and carry out the required arithmetic.

7 INTERTEMPORAL MODELS

Intertemporal (that is, dynamic) models are ones in which it is possible to report time paths of endogenous variables. More formally, they are models in which one or more of the equations relates variables at two or more different time instants as in, for example, a capital accumulation equation

$$K(t+1) = K(t).D + I(t)$$

relating capital K(t+1) at time t+1 to capital K(t) at time t, the depreciation rate D and investment I(t) during period t.

Models framed in continuous time must be made discrete by selecting a finite set of time instants over which to solve them. As part of this, any derivatives with respect to time are usually replaced by a suitable finite difference; for example, the derivative $f'(t)$ of f(t) with respect to time t may be replaced by

$$[f(t+1) - f(t)]/Y(t)$$

where Y(t) is the number of years between time instants t and t+1.

There are several methods available for solving such models. GEMPACK contains an implementation of the finite-difference simultaneous method described in Exercise 5.12 in Chapter 5 of Dixon *et al.* (1992). This involves solving the linearized equations at all time instants simultaneously at each step of the multi-step calculation.

The great virtue of this approach is that it is truly a general-purpose method which, for example, can just as easily handle forward-looking behaviour as backward-looking, and which works just as well when there are a large number of state variables. Many of the other methods require special user intervention or initial setting up depending on the forward/backward nature of the behaviour being modelled, and some are rather inefficient if there are more than a small number of state variables.

More details of the approach in GEMPACK can be found in Codsi *et al.* (1992).

8 COMMUNICATING MODELS TO OTHERS

GEMPACK contains tools which make it easy to move models between different machines (that is, ones with different operating systems, such as a PC and a mainframe) on which GEMPACK is installed. This can enable other modellers to independently check results and try alternative scenarios, and encourages modellers to open up their models to scrutiny by others (rather than keeping them as black boxes).

It is easy to move text files between different computers using utilities such as Kermit, FTP and Apple File Exchange. However, as a general rule, binary files cannot be moved easily from one operating system to another.

The essential ingredients of a model are

(1) the TABLO Input file,

(2) the data file(s), and

(3) any relevant Command files for specifying the closure or for carrying out simulations, and/or any Stored-input files for condensing the model.

Of these, (1) and (3) are text files (hence easily transferred) but the data files are often binary files (the Header Array files in GEMPACK, described in section 6 above).

To move a Header Array data file from one machine to another, first run the GEMPACK program RWHAR on the first machine; this converts it to a text file. Then transfer this text file to the second machine and, on that machine, run the GEMPACK program MKHAR; this converts the text file to a Header Array data file.

Once all the files for a particular model have been transferred in this way, the model can be solved on the second machine. Since the TABLO Input file contains a complete description of the theory of the model, this also enables the modeller on the second machine to look in detail at the model (and perhaps suggest modifications or additions).

9 SOFTWARE ASPECTS

In this section we describe a few features of the software design. Readers not interested in this topic may prefer to skip to section 10.

9.1 Standard Program Options

All GEMPACK programs have options which allow users either to take input from a file (we call them Stored-input files), or direct output to a Log file. Of course, many machines have operating-system-dependent ways of doing this. The GEMPACK options provide an operating-system-independent means which looks the same on all machines.

When the programs are run interactively, they allow recovery from invalid input. They also provide an option BAT which, if selected, indicates that the program should stop with an error message if invalid input is detected; this option can be selected if the program is being run in batch mode to protect against the possibility of spurious results that might otherwise be produced if the program were to allow recovery from this invalid input.

9.2 Subroutines

The source code of GEMPACK consists of about 20 main programs and several hundred subroutines. The code has been designed to collect any non-portable aspects (for example, file naming, opening and closing, and different record-length limits) into a small number of subroutines, with the rest being identical on all machines. Then, when GEMPACK is ported to a new machine, just these few non-portable subroutines need to be modified and tested. On machines with a source-code licence (see section 11 below), typically one of the first steps in the installation is to build a library containing object modules for all of the subroutines.

Except in some of the non-portable subroutines, all code has been written strictly according to the 1977 Ansi Fortran standard. The only deviations from this have been in the GEMSIM- and TABLO-specific routines where we have used "INCLUDE" statements to include declarations of many arrays in COMMON.

9.3 Model Size And Program Parameters

One of the problems in providing a general-purpose suite of software is that of adjusting the programs to handle models of different sizes (requiring differing amounts of memory). GEMPACK has taken a fairly simple approach to this problem. The main idea is that the library of subroutines (which forms the vast bulk of the GEMPACK source code) should never need to be recompiled when memory requirements increase.

All arrays whose size may be model-dependent (for example, an array holding the names of the variables of the model) are declared in the main program and passed down to any subroutine requiring them. The sizes of these arrays are declared using Fortran PARAMETERs. These sizes are passed down, as well as their names (which are passed as CHARACTER strings). When data is added to such an array, the routine checks that the array is large enough. If not, it sends a message saying which main program PARAMETER must be increased (this is why the name must be passed down) and by how much. The user must then edit the main program to increase the size of the relevant PARAMETER, then recompile the main program and link it to the subroutine library.

Note that, under this strategy, even work arrays required at a relatively deep level in the subroutine calling chain must be declared in the main program and passed down. This makes calling sequences relatively complicated, but has the advantage that subroutines never need to be recompiled, just main programs.

It also precludes putting model-dependent arrays in COMMON since then their size would need to be altered in the source-code of the subroutines as well as in the main program. The use of Include files (as we have done with TABLO and GEMSIM) gets around this. However this does produce slightly non-portable code and causes some file-management problems for users; accordingly we have restricted the use of Include files to TABLO and GEMSIM.

One consequence of this is that modellers require access to the source code of GEMPACK and to a Fortran compiler if they are to be able to reconfigure the

programs to handle larger models. This is why we send the source code with our main versions of GEMPACK. (With the Executable Image version described in section 11, programs cannot be reconfigured for larger models.)

9.4 History Of Files

In a busy modelling outfit, one of the problems is that of keeping track of different versions of files (for example, TABLO Input files and data files) and simulations (different shocks and/or closures). As an aid to this, whenever GEMPACK programs create a new file, they automatically put the time and date on the file and also the name of the program used. This information is usually reported when the file is accessed subsequently.

In addition, some of the programs allow (even encourage) users to add so-called History or "verbal descriptions". These are stored on the relevant files as character data and are echoed when the file is accessed.

9.5 Binary Files

GEMPACK has always been designed with large models in mind. Since these typically have quite large data files, GEMPACK encourages the use of a type of binary file (a Header Array file - see section 6) which takes much less disk space than the corresponding text file, although text files are also allowed. Many other files (for example, Solution files holding simulation results) are also binary files. One disadvantage of having binary files is that utility programs must be provided to access them; for example, there are utility programs for displaying and modifying the data on binary Header Array files.

The software carrying out multi-step simulations creates several work files which are binary files to keep the disk requirements as small as possible. This is all done in a way which is essentially transparent to users.

Communication between different programs is often via files, which are usually binary files. Since these binary files do not need to be edited by users, they can contain information which is used by other programs to ensure the integrity of the information on them; this can provide useful extra checks in a complicated modelling situation. Standard file-name suffices are used to distinguish different types of files.

9.6 No Special Windows Features

None of the programs has windows-type features, such as pull-down menus. They all present essentially a sequential text window to a user. One advantage of this is that the software looks very similar under different operating systems.

Some of the programs must do heavy-duty number crunching and many of them require considerable, sometimes complicated, user input (for example, the information required for carrying out a simulation). Hence they are most often run in a batch-type mode by which we mean that the information required to run them is prepared in advance. With a large model, it would be a disadvantage rather than an advantage to have pull-down menus to specify, for example, the closure or shocks.

We have concentrated on providing interfaces such as GEMPACK Command files (see section 3.2) which are self-documenting and relatively easy to understand. We have also tried to make the programs relatively flexible by

providing options choices at the start of each program; most users are satisfied with the standard options, but more sophisticated users, or users with a particularly complex task, can take advantage of these options. On-line help is provided for these options.

9.7 Solving Sparse Systems Of Linear Equations

GEMPACK's ability to solve large models relatively efficiently is due in no small measure to the efficiency of the Harwell sparse linear-equations-solving routines MA28 written by Iain Duff (see Duff (1977)). MA28 is just one of the large number of general-purpose routines in the Harwell Subroutine Library which can be used to carry out a wide range of numerical calculations (including matrix calculations, solving differential equations, statistical calculations, numerical integration, root finding, and so on). More information about the software in this library and be obtained from

Harwell Subroutine Library
AEA Technology Harwell Laboratory
Oxfordshire OX11 0RA, England

9.8 Compilers On 386/486 PCs And Macintosh PCs

With a source-code version of GEMPACK a suitable Fortran compiler is required. At present,

- on a DOS 80386/80486 PC either the Lahey compiler F77L-EM/32 (version 5 or later) or the Watcom compiler Fortran 77/32 is required.
- on a Macintosh PC either of Absoft's compilers MacFortran/020 version 2.4 or MacFortran II (version 3.2 or later) is required.

10 CHANGES IN COMPUTING ENVIRONMENT

When GEMPACK development was begun in earnest (namely around 1985), nearly all modelling was done on mainframes; at the time PCs didn't have enough memory or sufficiently good Fortran compilers to do serious modelling. Now, of course, things are quite different. For example, there are some excellent Fortran compilers on PCs and 66Mhz 80486 PCs are about as fast as many of the readily available traditional modelling work-horses (such as relatively recent VAX/VMS machines). For example, the modellers at Monash University's Centre of Policy Studies and Impact Project (where very large models are the norm) have recently switched from the university's VAX/VMS to such PCs; they are using PCs with 48Mb of RAM and getting elapsed times on their PCs which are almost the same as CPU times on the VAX (and, since they would have to share the VAX with about 100 users during the day, many times less than the elapsed time would be on the VAX).

Users at other GEMPACK sites have switched to workstations, including, quite recently, ones with Alpha chips.

Because the GEMPACK software is essentially unchanged between different machines, and because models and data can be moved easily from one machine to another (see section 8 above), it has turned out to be relatively easy for users to change from one machine to another, or even to be working partly on a PC and

partly on a mainframe or workstation. This is perhaps one advantage of the relatively conservative software design employed for GEMPACK.

11 DIFFERENT VERSIONS OF GEMPACK

GEMPACK is sent to users either as source code or as executable images. An introduction to the different versions is given below. The Source-code and Executable Image versions come with full user documentation (approximately 400 pages - see section 11.4).

11.1 Source-code Versions

Prior to Release 5.1, all recent versions of GEMPACK were source-code versions. With these versions, a suitable Fortran compiler is required. The size of models that can be handled is limited only by the amount of memory on the computer on which the software is installed. The source code of the full GEMPACK occupies about 5 megabytes of disk space, and several extra megabytes of diskspace are required to produce executable images of the programs and to build and/or modify models.

Source-code versions are currently available for 80386/80486 PCs running DOS (or Windows or OS/2), Macintosh PCs, VMS (VAX and DEC Alpha) and Unix machines. Other machines may be added in the future.

11.2 Executable Image Version

The Executable Image version of GEMPACK is available for 80386/80486 PCs running DOS (or Windows or OS/2). It consists of executable images of the most commonly-used GEMPACK programs. No Fortran compiler is required in this case. Models are limited in size by the configuration of the programs as sent. The standard Executable Image Version runs on a machine with 8 megabytes of memory. It can handle moderately large models, including all those in section 2.6. Because no source code is sent with this version, the programs cannot be reconfigured to handle larger models.

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 standard Executable Image Version runs on 80386/80486 PCs with at least 8 megabytes of memory, a numeric coprocessor and a hard disk.

11.3 Demonstration Version

The Demonstration Version is very similar to the Executable Image Version of GEMPACK except that it is restricted to small models. It is intended for essentially free distribution so potential users can assess the capabilities of GEMPACK.[14] We also expect it will be useful in teaching situations. 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. This

---

14 Copies of the Demonstration version can be obtained by sending 20 Australian dollars to the GEMPACK Manager at the Impact Project, Monash University, Clayton 3168, Australia. If sending from overseas, this must be in the form of a bank draft in Australian dollars which is payable on an Australian bank. You can obtain such a draft from your local bank.

version can handle all the models in section 2.6 except for ORANI-F, TRADMOD and the 5x6 and 10x7 versions of GTAP. The Demonstration Version runs on DOS 80386/80486 PCs with at least 4 megabytes of memory, a numeric coprocessor and a hard disk.

11.4 Current GEMPACK User Documentation

GPD-1, *An Introduction to GEMPACK*, Second edition, April 1994, pp.252+15.

GPD-2, *User's Guide to TABLO, GEMSIM and TABLO-generated Programs*, Second edition, April 1994, pp.138+14.

GPD-3, *How to Create and Modify GEMPACK Header Array Files Using the Program MODHAR*, Third edition, April 1993, pp.27+4.

APPENDIX A

THE TABLO INPUT FILE FOR THE STYLIZED JOHANSEN MODEL

We begin this appendix with the full TABLO Input file for Stylized Johansen. The discussion of this file, which was begun in section 2.3.1 of the main paper, is continued at the end of the file.

```
!------------------------------------------------------------------------!
!                    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 ! ;
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))] ;
```

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

Notes on the TABLO Input file for Stylized Johansen

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 TABLO Input file is not case-sensitive so, for example, XH and Xh would be identical so far as TABLO is concerned.

Defaults

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 users to reduce the number of qualifiers required in your TABLO Input files. After the statement

```
VARIABLE (DEFAULT = LEVELS) ;
```

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 a TABLO Input file are linearized ones, the opposite default statement

```
EQUATION (DEFAULT = LINEAR) ;
```

can be added near the start of the file, and then only levels equations would need to be flagged, using the qualifier (LEVELS).

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 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.

Sets

Next come the declarations of the SETs, 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.

Variables

Then come the declarations of the VARIABLEs. Note that the arguments (if any) of each are clearly described, using the "(all,<index>,<set-name>)" quantifier(s) at the start of the declaration. 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.

Coefficients

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.

File

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 the computer containing this data is not limited by this logical name; the actual file can be given any convenient name. GEMSIM or the TABLO-generated program will prompt for this actual name when run; the prompt will use the logical name 'iodata' from the TABLO Input file. Or, in a GEMPACK Command file, the logical name is linked to the actual name by the relevant statement (for example, "file iodata = sj.dat ;")

Reads

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).

Formulas

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.

You may notice that there is no FORMULA assigning an initial value to the levels variable Y (nominal household expenditure). This is because this variable does not appear in any of the linearized EQUATIONs. (The only EQUATION in the TABLO Input file involving Y is the linear EQUATION "Consumer_demands" which has the linear variable p_Y in it, but not Y itself.) Thus it is not necessary to give a FORMULA for the initial value of Y. [Indeed, if a FORMULA for Y was added, TABLO would indicate this seems to be redundant because Y does not appear in the system of linearized equations.]

Equations

Finally come the EQUATIONs (see (E1) to (E10) in section 3.1.1 above). 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.

Displays and Writes

Note also that there are DISPLAY and WRITE statements to enable users 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. The following statements could be added at the end of the TABLO Input file for Stylized Johansen.

```
DISPLAY ALPHACOM ;
WRITE ALPHAFAC TO TERMINAL ;
FILE (NEW, TEXT) output ;
WRITE ALPHACOM TO FILE output ;
WRITE ALPHAFAC TO FILE output ;
```

[These WRITE features plus TABLO's ability to process FORMULAs give TABLO some of the properties of a database manipulator. This can be used when processing data files.]

APPENDIX B

INITIAL VALUE PROBLEMS

In this appendix, we show formally how the sorts of simulation problems that GEMPACK is designed to solve can be converted to Initial Value problems, as stated in section 4.3.1. The Simulation problems are defined in section B.1 while the conversion to an Initial Value problem is given in section B.2.

GEMPACK solves Simulation problems by using Euler's method, the midpoint-method or Gragg's method to solve the corresponding Initial Value problem. Section B.3 indicates how the relevant numerical partial derivatives are calculated from the symbolic equations in the TABLO Input file and the pre-simulation data base.

More details about the use of these methods and extrapolation in GEMPACK is given in Pearson (1991).

B.1 Simulation Problems - A Definition

GEMPACK can be used to solve those economic models which can be written as a finite number of equations[15]

$$g_i(y_1, \ldots, y_{n+m}) = 0 \quad i=1, \ldots, m$$

where $g_1, \ldots, g_m$ are functions of the m+n variables $y_1, \ldots, y_{n+m}$. To solve this model, n of the variables must be specified exogenously leaving the other m endogenous. We use $x_1, \ldots, x_n$ to denote a set of exogenous variables and $z_1, \ldots, z_m$ the corresponding endogenous ones. Then the equations of the model can be written as

$$g_i(z_1, \ldots, z_m, x_1, \ldots, x_n) = 0 \quad i=1, \ldots, m. \tag{1.1}$$

(The m used in this appendix is the same as *m* in section 4.1 while *p* in section 4.1 equals m + n.)

It is convenient to introduce vectors (we use bold-face type for these)

$$\mathbf{z} = (z_1, \ldots, z_m)$$

$$\mathbf{x} = (x_1, \ldots, x_n)$$

$$\mathbf{g}(\mathbf{z}, \mathbf{x}) = (g_1(\mathbf{z}, \mathbf{x}), \ldots, g_m(\mathbf{z}, \mathbf{x})).$$

Here $\mathbf{g}$ is a vector-valued function from $\mathbf{R}^{n+m}$ to $\mathbf{R}^m$ and the equations of the model are

$$\mathbf{g}(\mathbf{z}, \mathbf{x}) = \mathbf{0}. \tag{1.2}$$

Because $x_1, \ldots, x_n$ is a set of exogenous variables, the equations $\mathbf{g}(\mathbf{z}, \mathbf{x}) = \mathbf{0}$ determine $\mathbf{z}$ as a function of $\mathbf{x}$, say $\mathbf{z} = \mathbf{f}(\mathbf{x})$. This means

$$z_i = f_i(\mathbf{x}) = f_i(x_1, \ldots, x_n) \quad i = 1, \ldots m$$

where[16]

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})).$$

---

15 Models whose underlying theory involves inequalities as well as equations are not covered unless the inequalities can be rewritten as equations.

16 In practice, although the functions $g_1, \ldots g_m$ are known (in the sense that formulae for them are given), explicit formulae for $\mathbf{f}$ or $f_1, \ldots, f_m$ are not known; there is merely a guarantee of their existence. (If an explicit formula for $\mathbf{f}$ was known, the model would be solved easily by substituting the values of the exogenous variables x_j into this formula.)

It is assumed below that each function f_i has continuous first partial derivatives at each point **x** in some relevant domain and also that each function g_i has continuous first partial derivatives with respect to each of its m + n variables. That is, it is assumed that

$$\frac{\partial g_i}{\partial z_j} \quad \text{and} \quad \frac{\partial g_i}{\partial x_k}$$

exist and are continuous functions of **z** and **x** at each point in some relevant domain.

By a **Simulation problem** for such a model we mean that one solution of the model, say $\mathbf{z} = \mathbf{z}_0$ where $\mathbf{x} = \mathbf{a}$, is given and also given is another set of values for the exogenous variables, say $\mathbf{x} = \mathbf{b}$. The problem is to calculate the value $\mathbf{z}_1$ of **z** when $\mathbf{x} = \mathbf{b}$. That is,

$$\left.\begin{array}{l}\text{given } \mathbf{z} = \mathbf{z}_0 \text{ when } \mathbf{x} = \mathbf{a} \text{ is a solution of} \\ \text{(1.1) or (1.2), find the value of } \mathbf{z}_1 \text{ of } \mathbf{z} \text{ when } \mathbf{x} = \mathbf{b}.\end{array}\right\} \qquad (1.3)$$

In the notation used above,

$$\mathbf{g}(\mathbf{z}_0 , \mathbf{a}) = \mathbf{0} \quad (\text{or } \mathbf{f}(\mathbf{a}) = \mathbf{z}_0) ,$$

and the problem is to find $\mathbf{z}_1$ such that

$$\mathbf{g}(\mathbf{z}_1 , \mathbf{b}) = \mathbf{0} \quad (\text{or } \mathbf{f}(\mathbf{b}) = \mathbf{z}_1) \ .$$

We let

$$\mathbf{a} = (a_1 ,...,a_n) \text{ and } \mathbf{b} = (b_1 ,...,b_n).$$

B.2 Converting a Simulation Problem to an Initial Value Problem

The key is to introduce a new scalar variable v (a real number) and to consider the exogenous variables **x** as a function of v given by

$$\mathbf{x} = \mathbf{a} + v(\mathbf{b} - \mathbf{a}).$$

Note that $\mathbf{x} = \mathbf{a}$ when $v = 0$, $\mathbf{x} = \mathbf{b}$ when $v = 1$ and that x moves along the straight line (in n dimensions) joining **a** and **b** as v increases from 0 to 1.

Because **x** is a function of v and **z** is a function of **x** via $\mathbf{z} = \mathbf{f}(\mathbf{x})$ or $\mathbf{g}(\mathbf{z}, \mathbf{x}) = \mathbf{0}$, it follows that **z** is also a function of v. Indeed,

$$\mathbf{z} = \mathbf{f}(\mathbf{x}) \text{ where } \mathbf{x} = \mathbf{a} + v(\mathbf{b} - \mathbf{a})$$

or, alternatively,

$$z_i = f_i(\mathbf{x}) \quad \text{for } i = 1, ..., m$$

where

$$x_k = a_k + v(b_k - a_k) \text{ for } k = 1, ...,n.$$

By assumption each function f_i has continuous first partial derivatives while clearly each x_j is a differentiable function of v. Thus, by the Chain Rule (see, for example, Theorem 1 in section 8.7 of Kreyszig (1979)), each z_i is a differentiable function of v and

$$\frac{dz_i}{dv} = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} \frac{dx_j}{dv} = \sum_{j=1}^{n} \frac{\partial z_i}{\partial x_j} \frac{dx_j}{dv} \quad .$$

However, because no formulae for the functions $f_1, \ldots, f_m$ are given (there is merely a guarantee of their existence), the formula above does not give an effective way of calculating $\frac{dz_i}{dv}$. But we can go back to the original equations of the model to get an effective way of calculating these derivatives. Recall from (1.1) that

$$g_i(z_1,\ldots,z_m, x_1,\ldots,x_n) = 0 \quad \text{for } i = 1,\ldots,m \ .$$

By assumption $\frac{\partial g_i}{\partial z_j}$ and $\frac{\partial g_i}{\partial x_k}$ exist and are continuous functions of **z** and **x**. Also we have seen that each z_j is a differentiable function of v (as is each x_k). Thus the Chain Rule can be used to differentiate the above equations with respect to v. This gives[17]

$$\sum_{j=1}^{m} \frac{\partial g_i}{\partial z_j} \frac{dz_j}{dv} + \sum_{k=1}^{n} \frac{\partial g_i}{\partial x_k} \frac{dx_k}{dv} = 0 \text{ for } i = 1, \ldots, m. \qquad (2.1)$$

But $x_k = a_k + v(b_k - a_k)$ so that

$$\frac{dx_k}{dv} = b_k - a_k \text{ for } k = 1,\ldots,n.$$

Thus, expressing (2.1) in matrix form gives

$$\begin{bmatrix} \frac{\partial g_1}{\partial z_1} & \cdots & \frac{\partial g_1}{\partial z_m} \\ \vdots & & \vdots \\ \frac{\partial g_m}{\partial z_1} & \cdots & \frac{\partial g_m}{\partial z_m} \end{bmatrix} \begin{bmatrix} \frac{dz_1}{dv} \\ \vdots \\ \frac{dz_m}{dv} \end{bmatrix} = - \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & & \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_m}{\partial x_n} \end{bmatrix} \begin{bmatrix} b_1 - a_1 \\ \vdots \\ b_n - a_n \end{bmatrix} .$$ That

is,

$$E(\mathbf{z}, \mathbf{x}) \frac{d\mathbf{z}}{dv} = D(\mathbf{z}, \mathbf{x})$$

---

17 The equation (2.1) is essentially the equation $C\mathbf{z} = 0$ in section 4.1. The matrix C in section 4.1 corresponds to the m×(m+n) matrix whose entries are $\partial g_i / \partial z_j$ in row i and column j, and $\partial g_i / \partial x_k$ in row i and column (m+k). The vector **z** in section 4.1 is the (m+n)×1 vector whose first m components are dz_i / dv (i=1, ..., m) and whose last n components are dx_k / dv (k=1, ..., n).

where $\mathbf{x} = \mathbf{a} + v(\mathbf{b} - \mathbf{a})$, $E(\mathbf{z}, \mathbf{x})$ is the $m \times m$ matrix whose (i, j) entry is $\frac{\partial g_i}{\partial z_j}$ evaluated at $(\mathbf{z}, \mathbf{x})$, $\frac{d\mathbf{z}}{dv}$ is the $m \times 1$ vector whose jth entry is $\frac{dz_j}{dv}$ and $D(\mathbf{z}, \mathbf{x})$ is the $m \times 1$ vector obtained by multiplying the $m \times n$ matrix whose (i, k) entry is $\frac{\partial g_i}{\partial x_k}$ evaluated at $(\mathbf{z}, \mathbf{x})$ by the $n \times 1$ vector whose kth entry is $a_k - b_k$. Now $\mathbf{x} = \mathbf{a} + v(\mathbf{b} - \mathbf{a})$ depends on v. If we let

$$A(\mathbf{z}, v) = E(\mathbf{z}, \mathbf{a} + v(\mathbf{b} - \mathbf{a})),$$
$$B(\mathbf{z}, v) = D(\mathbf{z}, \mathbf{a} + v(\mathbf{b} - \mathbf{a}))$$

then, for all relevant v,

$$A(\mathbf{z}, v)\frac{d\mathbf{z}}{dv} = B(\mathbf{z}, v) . \qquad (2.2)$$

For a given $\mathbf{z}$ and v, this is a system of m linear equations in the m unknowns $\frac{dz_1}{dv}, \ldots, \frac{dz_m}{dz}$. Indeed this is essentially equation (3), $A.\mathbf{z_1} = \mathbf{b}$, of section 4.1 in a different notation. The matrix A in section 4.1 is what is called $A(\mathbf{z}, v)$ here, the vector $\mathbf{b}$ in section 4.1 is what is called $B(\mathbf{z}, v)$ here, and the vector $\mathbf{z_1}$ in section 4.1 is essentially the same as $\frac{d\mathbf{z}}{dv}$ here.

In practice, it seems to be the case that the matrix $A(\mathbf{z}, v)$ is invertible for the relevant values of $\mathbf{z}$ and v occurring for a valid closure in a well-specified model.[18] In what follows we assume that $A(\mathbf{z}, v)$ is invertible at all relevant points. Then, for any relevant $\mathbf{z}$ and v, we can solve (2.2) and so calculate $\frac{d\mathbf{z}}{dv}$ (in terms of $\mathbf{z}$ and v).[19]

Thus the simulation problem has been converted to the following problem (in the sense that, if we can solve this problem, we can solve the simulation problem).

$$\left.\begin{array}{l} \text{Given } \mathbf{z} = \mathbf{z}_0 \text{ when } v = 0, \text{ and} \\ \text{that } \frac{d\mathbf{z}}{dv} \text{ can be calculated by solving (2.2).} \\ \text{Find } \mathbf{z}_1, \text{ the value of } \mathbf{z} \text{ when } v = 1. \end{array}\right\} \qquad (2.3)$$

---

18 We do not know of a theoretical result which guarantees the invertibility of this matrix. This is a question that needs further investigation. (The only examples we are aware of where A(z, v) is not invertible at some point are cases where $\mathbf{g}(\mathbf{z}, \mathbf{x}) = 0$ does not have a unique solution for z in terms of x in a neighbourhood. This is not unlike the example in equation (35.6) of Dixon *et al.* (1982).)

19 As explained in section 35.4 of Dixon *et al.* (1982), we do not know *ex ante* whether (2.2) can be solved but numerical calculations can leave us confident *ex post*. (Also, while there is a very clear theoretical distinction between invertible and non-invertible matrices, the distinction is blurred in practice because of rounding errors in any actual calculation.)

The problem in (2.3) is, of course, an example of an Initial Value Problem; these problems are well-known (they occur in many areas) and widely studied.

An **Initial Value Problem** is a problem of the following form. Consider a vector $\mathbf{w} = (w_1, \ldots, w_m)$ depending on a real scalar variable u such that **w** is a differentiable function of u.[20] Given is a function **q**(**w**, u) of **w** and u such that

$$\frac{d\mathbf{w}}{du} = \mathbf{q}(\mathbf{w}, u)$$

for all **w** and u in some suitable domain.[21] Also given is the value $\mathbf{w}_0$ of **w** when $u = u_0$ (the "initial values") and another value, say u_1, of u. The problem is to calculate the value $\mathbf{w}_1$ of **w** when $u = u_1$. That is,[22]

$$\left.\begin{array}{l} \text{given } \mathbf{w}_0 = \mathbf{w}(u_0), \text{ and} \\ \quad \dfrac{d\mathbf{w}}{du} = \mathbf{q}(\mathbf{w}, u) \text{ for all relevant } \mathbf{w}, u, \\ \text{the problem is to calculate } \mathbf{w}_1 = \mathbf{w}(u_1). \end{array}\right\} \qquad (2.4)$$

B.3 Setting up the Initial Value Problem

The initial solution $\mathbf{z}_0$ (when v = 0), in the notation of (2.3) above, is inferred from the READs and FORMULAs in the TABLO Input file.

The values of the partial derivatives $\frac{\partial g_i}{\partial z_j}$ are calculated by substituting the relevant values of z_j, a_j, b_j and v into symbolic expressions obtained directly from the algebraic form of the linearized equations - either those on the TABLO Input file (if the linearized equation is explicitly on this file) or as obtained by symbolically differentiating any levels equations on the TABLO Input file. The system (2.2) of linear equations is solved using the Harwell sparse linear equations solving routines MA28 (see section 9.7).

Because the cost of forming up and of solving (2.2) is, in general, quite high, it would not be suitable to use an Initial Value solution method requiring a relatively large number of function evaluations; Euler, midpoint and Gragg require relatively few function evaluations.

---

20 That is, each w_i is a differentiable function of u.

21 $\frac{d\mathbf{w}}{du}$ is the vector $(\frac{dw_1}{du}, \ldots, \frac{dw_m}{du})$ and **q** is a function from $\mathbf{R}^{m+1}$ into $\mathbf{R}^m$.

22 A simple example (in which m=1) is: given $\frac{dw}{du} = w$ and $w = 1$ when $u = 0$, find w when $u = 1$. The solution is $w = e^u$ (since $w = e^u$ in general).

As explained in section 3.5.4 of Dixon *et al.* (1982), we usually do not know *ex ante* whether (2.2) can be solved. Nor do we usually have any *ex ante* guarantee that the theoretical conditions known to guarantee the existence of a unique solution to the Initial Value problem, or to guarantee that the method used will converge to this solution, will hold. But numerical calculations can leave us confident *ex post* that we are converging to a solution.

REFERENCES

Adams, P.A., P.B. Dixon, D. McDonald, G.A. Meagher and B.R. Parmenter (1993) 'Forecasts for the Australian Economy Using the MONASH Model', *Paper presented to International Symposium on Economic Modelling, University of Piraeus, Greece.*

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

Borrell B., D. Quirke, Beulah de la Peña and Lourdes Novena (1994) 'Philippine Sugar: The Industry Finding Its Feet', Centre for International Economics, Canberra.

Brooke, Anthony, David Kendrick and Alexander Meeraus (1998) *GAMS: A User's Guide*, The Scientific Press, Redwood City.

Centre for International Economics (1992) 'Relative Assistance for Export and Import Competing Industries', prepared for Sri Lankan Export Development Board, Canberra.

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.

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.

Dee, P.S. (1989) 'FH-ORANI: A Fiscal ORANI with Horridge Extension', *Impact Project Working Paper*, No. OP-66, pp. 367.

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.

Gao, X., (1993) 'China's Foreign Exchange Rate Regime and Its Impact on Exports and Growth', PhD Dissertation, National Centre for Development Studies, Australian National University, Canberra.

Harrigan, F. (1993) 'Software Reviews: Software for Solving Numerical General Equilibrium Models.', *The Economic Journal*, vol.103, pp1088-1104.

Harrison, W.J., K.R. Pearson, A.A. Powell and E.J. Small (1993) 'Solving Applied General Equilibrium Models Represented as a Mixture of Linearized and Levels Equations', *Impact Preliminary Working Paper* No. IP-61, Monash University, Clayton (September 1993), pp.20. [A modified version is to appear in *Computational Economics*.]

Harris, D. and D. Pearce (1992) 'Review of Global Red Meat Markets', Centre for International Economics, Canberra.

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.

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.

Huang, Y. (1993) 'Government Intervention and Agricultural Performance in China', PhD Dissertation, Australia-Japan Research Centre, Australian National University, Canberra.

Hughes, H. (1990) 'Asian Interdependence: the Impact of Chinese Exports on Other Asian Economies', National Centre for Development Studies, Australian National University, Canberra.

James, M. and R. McDougall (1993) 'FIT:An Input-Output Data Update Facility for SALTER', *SALTER Working Paper,* No. 17, Canberra, Australia: Industry Commission.

Jomini, P., J.F. Zeitsch, R. McDougall, A. Welsh, S. Brown, J. Hambley and J.Kelly (1991) 'SALTER: A General Equilibrium Model of the World Economy, Vol. 1. Model Structure, Database and Parameters', Canberra, Australia: Industry Commission.

Kreyzig, Erwin (1979) *Advanced Engineering Mathematics,* 4th edition, Wiley, New York.

Mai, Y., (1993) 'The Role of Policy in Industrial Upgrading in Asian NIEs', PhD Dissertation, National Centre for Development Studies, Australian National University, Canberra.

Malakellis, M. (1992) 'An Intertemporal Applied General Equilibrium Model Based on ORANI', *Impact Project Working Paper,* No. OP-72, pp. 37

Martin, W. (1991) 'Effects of Foreign Exchange Reform on Raw Wool Demand: a Quantitative Analysis', in C. Findlay (ed.), *Challenges of Economic Reform and Industrial Growth: China's Wool War,* Allen and Unwin, Sydney.

McDougall, R.A. (1994) 'Implementation of a Ramsey Problem Model in GEMPACK', *Centre of Policy Studies/Impact Project,* Research Memorandum No. 9402.

National Centre for Development Studies (1990) 'An Economy-wide Model of Papua New Guinea: Theory, Data and Implementation', Australian National University, pp.100.

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.

Powell, Alan,A. (1988) 'Impact Project Report: A brief account of activities over the period 1st March 1985 to 31st December 1987, with a prospectus for further developments.', *Impact Project Report* No. R-07.

Press, W.H., B.P. Flannery, S.A. Tenkolsky and W.T. Vetterling (1986) *Numerical Recipes: The Art of Scientific Computing,* Cambridge University Press.

Quirke, D. and D. Vincent (1993) 'An Economy-wide Framework to Analyse Policy Initiatives in Zimbabwe's Economic Structural Adjustment Program', Centre for International Economics, Canberra.

Rutherford, T.F. (1989) *General Equilibrium Modelling with MPS/GE,* Department of Economics, University of Western Ontario.

Suphachalasai, S. (1989) 'The Effect of the Government Intervention and the Multifibre Arrangement on the Thai Clothing and Textiles Industry', PhD Dissertation, National Centre for Development Studies, Australian National University, Canberra.

Trewin, R., Erwidodo and Yiping Huang (1993) 'Stages of Development of an Indonesian CGE Model (INDOGEM) with Application to Analysis of Key Agricultural Policies', *1993 Conference of Economists, Murdoch University*.

Warr, P.G. and I.A. Coxhead (1993) 'The Distributional Impact of Technical Change in Philippines Agriculture: A General Equilibrium Analysis', *Food Research Institute Studies,* Vol XXII, No. 3, pp.253-274.

Wilcoxen, P.J. (1989) 'Intertemporal Optimization in General Equilibrium: A Practical Introduction', *Impact Preliminary Working Paper* No. IP-45, Melbourne (December), pp.170.

Woldekidan, B. (1993) 'The General Equilibrium Model of Papua New Guinea', *South Pacific Working Papers* SP 93/4, National Centre for Development Studies, Australian National University, Canberra.

Yang, Y., (1994) 'The Impact of MFA Phasing Out on World Clothing and Textile Markets', *Journal of Development Studies,* 30(4) (forthcoming).