Impact Project

Impact Research Centre, Eleventh Floor, Menzies Building
Monash University, Wellington Road, CLAYTON
Vic. 3168 AUSTRALIA
Telephone : (03) 565 5112 (from overseas: 61 3 565 5112)
Telex: AA 32691 Telegrams: MonashUni
Fax: (03) 565 5486 (from overseas: 61 3 565 5486)
Electronic mail: impact@vaxc.cc.monash.edu.au

HITCHHIKERS' GUIDE TO GEMPACK

by

Wayne CALDER

Industry Commission

Canberra

Computing Document No. C10-01 October 1992

ISSN 1031 9034 *ISBN 0 642 10217 1*

CONTENTS

Figure 4: Data manipulation

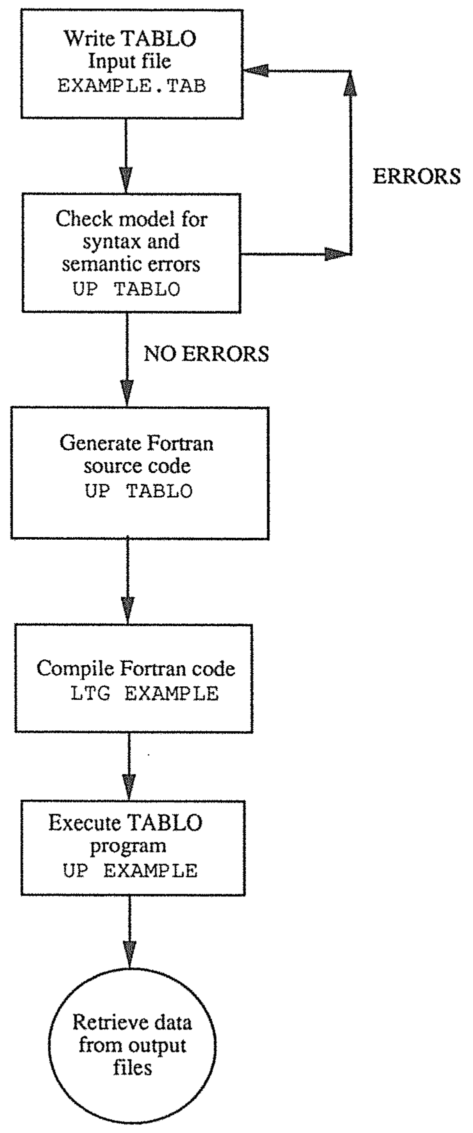
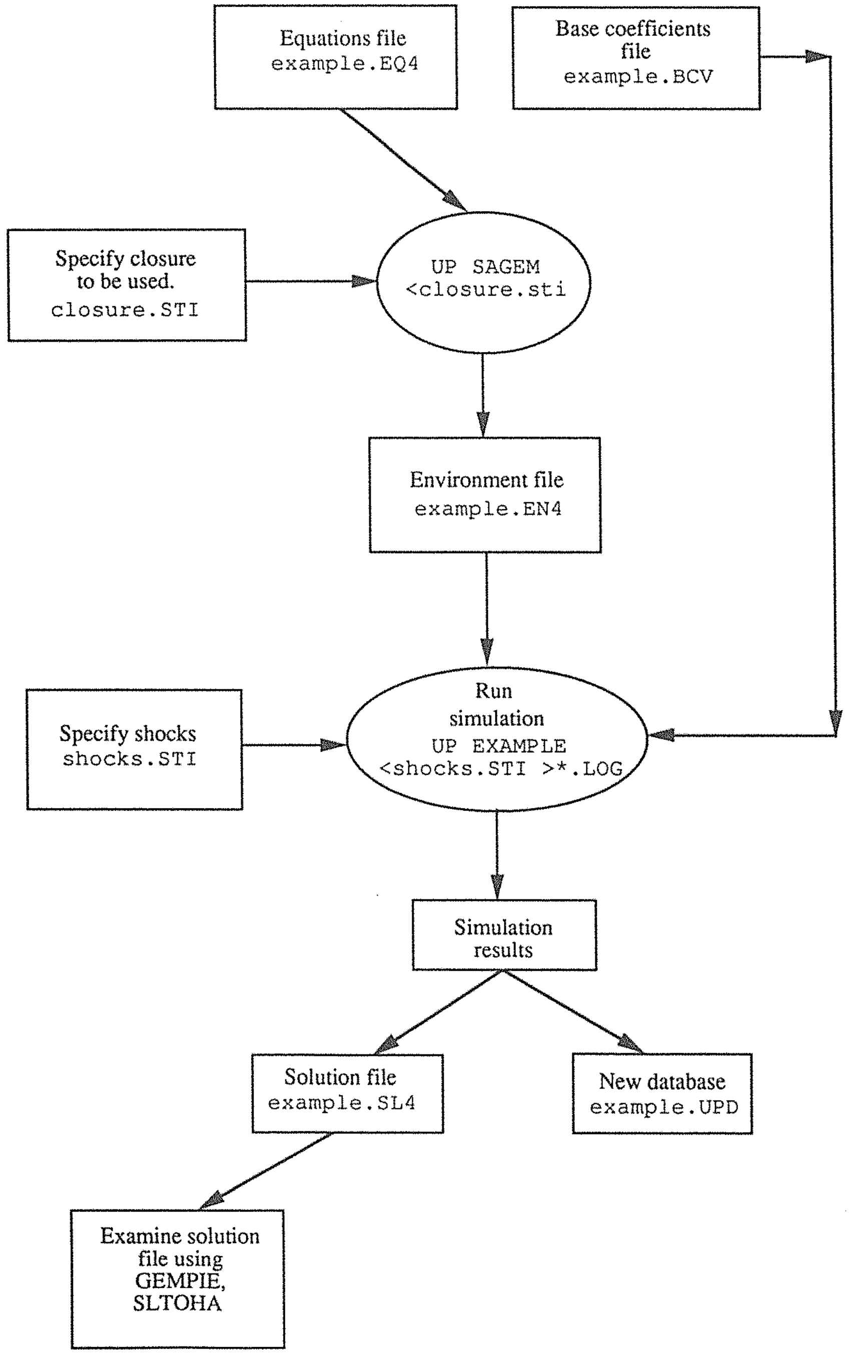

Write TABLO Input file EXAMPLE.TAB
ERRORS
Check model for syntax and semantic errors UP TABLO
NO ERRORS
Generate Fortran source code UP TABLO
Compile Fortran code LTG EXAMPLE
Execute TABLO program UP EXAMPLE
Retrieve data from output files

Figure 3: Running a simulation



ABSTRACT

GEMPACK (General Equilibrium Modelling Package) is a set of software designed to assist in the development and implementation of economic models, especially computable general equilibrium models. The software package has reduced the cost of model development by virtually eliminating the need for the modeller to write special purpose code and tailor–made programs. Users do not need to be familiar with any specialised computing languages because GEMPACK uses an essentially algebraic language which is easily understood and provides good documentation of the model itself.

This paper is a non–technical guide for first time users of GEMPACK. Firstly it describes the various components of GEMPACK in relation to developing and implementing economic models and using these models to run simulations. Secondly it covers the necessary practical details of exactly how the user will set up and use economic models to conduct simulations and the outputs that should be expected. Finally, it discusses how to use GEMPACK as a tool for manipulating data. A glossary of relevant GEMPACK terms is also included.

Figure 2: Model Implementation

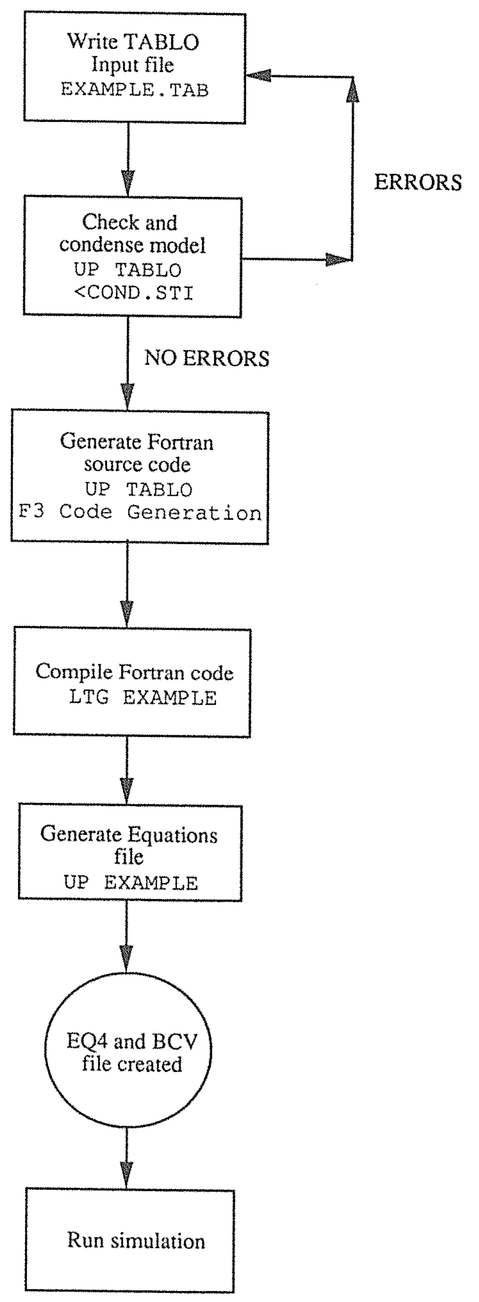
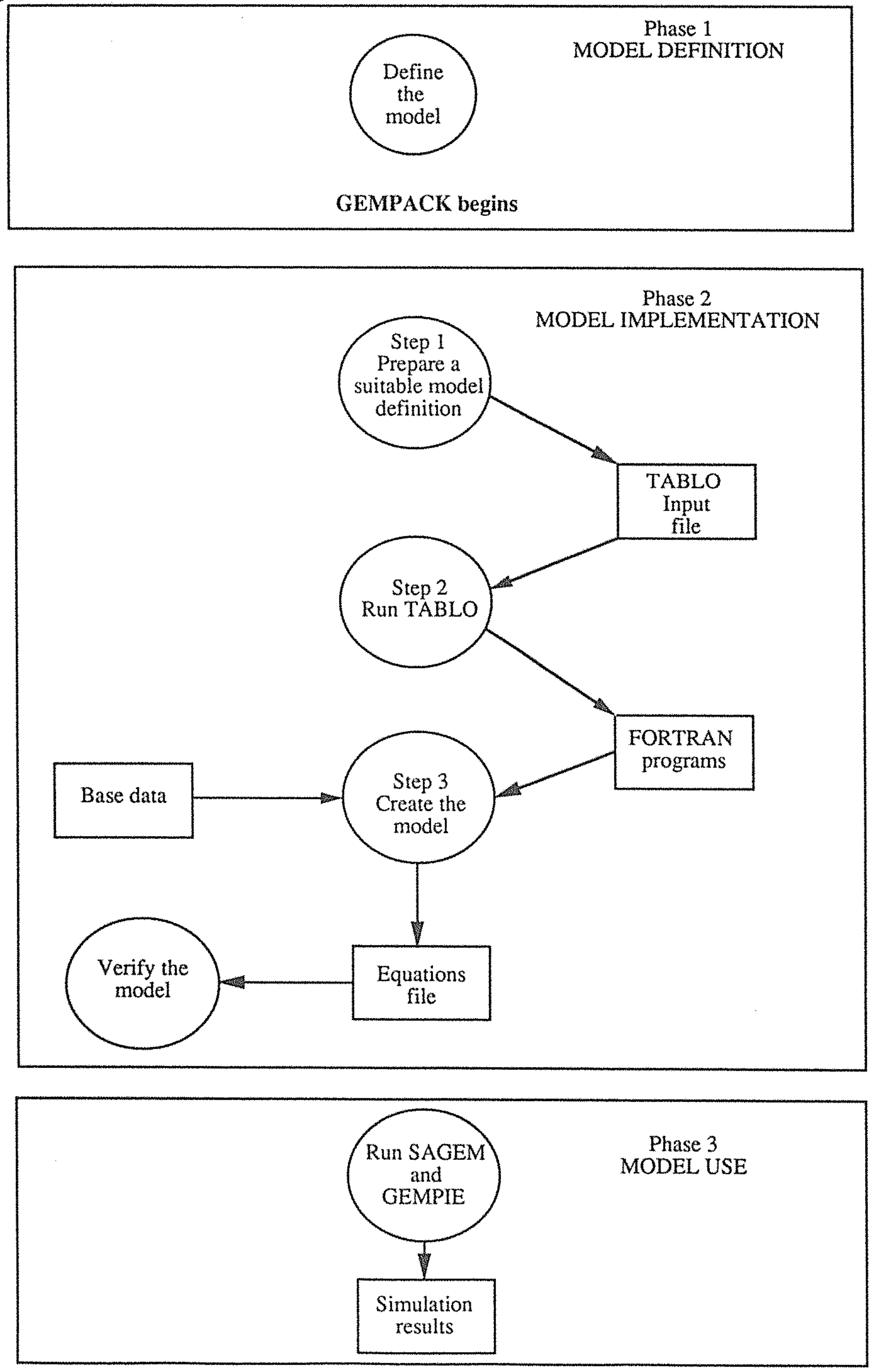


Figure 1: Overview of GEMPACK



Source: Codsi and Pearson 1988.

HITCHHIKERS GUIDE TO GEMPACK[1]

1 Introduction

The development and use of large computable general equilibrium (CGE) models by economists has become common in the last decade. Codsi and Pearson (1987) have gone as far as stating that it is the 'fastest growing area of applied economics'. The Industry Commission has used these models extensively for a number of years with the ORANI[2] and more recently the SALTER[3] and WEDGE[4] models.

As the name computable general equilibrium suggests, most of these models could not be used without computers. However, the computer implementation of such models has been a very time consuming and resource intensive activity. To understand why this is so, it is useful to first consider the stages that the model developer must go through when building an economic model.

As with any economic model a computable model requires a body of economic theory that captures those aspects of the economy that the modeller considers important. This procedure is called defining the model and consists of deriving the equations that capture the appropriate theory and assembling a database containing the behavioural and technological coefficients and the initial values of the variables.

In defining a model, a simplified version of the 'real world' is produced. This means it concentrates on certain aspects of the economy while abstracting from other features that are not viewed as vital to the problem being considered. In this way an economic model is analogous to a road map which, while concentrating on producing a scale version of the road system, abstracts from many other features of the terrain being followed.

After defining the model the user must write the necessary software before it can be implemented. This can be a time consuming task. Pearson (1988) reports that 48 person months were required to develop the original software for the ORANI model.

Along with being time consuming much of the special purpose code associated with CGE models is model dependent in that it is only relevant to a particular model and it is

[1] I would like to thank my colleagues at the Industry Commission, especially Michelle Gotch, Kevin Hanslow, Patrick Jomini and Ken Pearson of the Impact Project for their generous comments on earlier drafts. All remaining errors and omissions are mine.

[2] See Dixon, Parmenter, Sutton and Vincent 1982.

[3] See Jomini *et al* 1991.

[4] See Industry Commission Report No. 15.

difficult to incorporate relatively minor changes to its structure. Software is also often machine specific, designed for a particular computer and is not easily transportable to other machines.

These problems in model development make it desirable to devise a model independent software package which works equally well with a large number of models and which avoids the necessity of writing machine and/or model specific code. Also the package should be portable so that little effort is required to move between machines.

GEMPACK (General Equilibrium Modelling PACKage) is a set of general purpose software specifically designed to assist the implementation of large economic models, particularly computable general equilibrium models. GEMPACK consists of a number of programs that reduce the need for the modeller to be concerned with the computing details.

This document descirbes how to use Release 4.2.02 of GEMPACK. It is designed to give those people who have not used GEMPACK a general outline of the software and how to use it. Section two describes the individual components of GEMPACK in abstract while section three gives a step by step guide of how to use the package. This will give the reader the basic concepts behind GEMPACK as well as the necessary knowledge to use the package.

2 Overview of GEMPACK

GEMPACK consists of three main parts; one for handling data (MODHAR), one for model implementation (TABLO) and one for running simulations (SAGEM). This section will give a brief outline of each of these parts.

2.1 MODHAR (Modify a Header Array file)

The data used for running economic models with GEMPACK is held in files known as Header Array files. These files contain matrices of data each of which is identified by a four symbol identifier and can only be accessed by supplying the correct identifier.

Frequently in economic modelling it is necessary to modify the database to incorporate new information or completely change existing data. Unfortunately, Header Array files cannot be accessed by normal text editors as they are stored as binary files. The GEMPACK program MODHAR enables users to modify existing Header Array files or create completely new data files.

The GEMPACK publication 'Summary Documentation for MODHAR' provides a succinct explanation of MODHAR and how to use it, so no more will be said about it here.

Pearson, K. and Codsi, G. 1988, 'Stylized Johansen — An illustrative CGE model', Impact Project, GEMPACK document No. 23, first edition, Melbourne, June 1988.

Pearson, K. and Codsi, G. 1991a, 'The update and multi–step version of TABLO: User guide-lines', Impact Project, GEMPACK document No. 30, first edition, Melbourne, August 1991.

Pearson, K. and Codsi, G. 1991b, 'The update and multi–step version of TABLO: Syntax and semantic description', Impact Project, GEMPACK document No. 31, first edition, Melbourne, August 1991.

References

Codsi, G. and Pearson, K. 1986a, 'SAGEM users manual', Impact Project, GEMPACK document No. 3, first edition, Melbourne, May 1986.

Codsi, G. and Pearson, K. 1986b, 'User manual for GEMPIE and ORPIE', Impact Project, GEMPACK document No. 7, first edition, Melbourne, May 1986.

Codsi, G. and Pearson, K. 1986c, 'GEMPACK glossary', Impact Project, GEMPACK document No. 11, first edition, Melbourne June 1986.

Codsi, G. and Pearson, K. 1987, 'Influences of the application environment on the development of software for large economic models', Impact Project, Preliminary working paper No. IP–33, Melbourne, July 1987.

Codsi, G and Pearson, K. 1988a, 'A user's guide to TABLO Input files', Impact Project, GEMPACK document No. 24, first edition, Melbourne, June 1988.

Codsi, G. and Pearson, K. 1988b, 'GEMPACK: General–Purpose Software for Applied General Equilibrium and Other Economic Modellers', Computer Science in Economics and Management vol. 1, pp 189–207.

Codsi, G. and Pearson, K. 1988c, 'Update of Implementing economic models using TABLO', Impact Project, GEMPACK document No. 20a, first edition, Melbourne, February 1988.

Dixon, P., Parmenter, B., Sutton, J., and Vincent, D. 1982, *ORANI: A Multisectoral Model of the Australian Economy*, North–Holland, Amsterdam.

GEMPACK 1991, 'Summary documentation for MODHAR', Impact Project, Melbourne, February 1991.

IC (Industry Commission) 1991, Costs and Benefits of Reducing Greenhouse Gas Emissions, Report No. 15, AGPS, Canberra.

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

McDougall, R. 1990, 'Notes for lecture on model closure, ORANI training course' February 1990.

Pearson, K. 1988, 'Automating the Computation of Solutions of Large Economic Models', Economic Modelling, vol. 5, pp 385–395.

Pearson, K. and Codsi, G. 1987, 'Implementing economic models using TABLO', Impact Project, GEMPACK document No. 20, first edition, Melbourne, September 1987.

2.2 TABLO -- implementing a model[5]

Once the underlying theory of the model has been developed and the original system of equations (often nonlinear) describing this theory formulated, it is necessary to implement the model. The steps involved in implementation are shown in Figure 1. The implementation process consists of everything from transcribing the model definition into a suitable Input file to generating the Equations file. TABLO is the part of GEMPACK that handles model implementation.

The first step in implementing a model is to reformulate the original equation system into linear form. This is necessary as GEMPACK uses Johansen's linearisation procedure to calculate solutions (see Pearson and Codsi 1988). The Johansen method requires the equations of the original system to be approximated by linear equations whose variables are percentage changes in the levels of the original variables. Thus, the linearised version of the model is a first order approximation of the original non–linear system.

The Johansen method was chosen because of the relative ease of obtaining solutions to linear models compared with nonlinear models. This method does not give exact solutions because of the linearisation errors arising from expressing the underlying model as a set of linear equations. However, from the experience of the Impact Project and the Industry Commission thus far, it appears that the numerical results achieved are often of the correct order of magnitude making them accurate enough for economic policy interpretation. If the problem of linearisation error is too severe, the technique of multiple step simulation can be used to reduce the error. This procedure will be discussed in more detail later.

The next step is to transcribe the linearised version of the model into a form that can be processed by the GEMPACK program TABLO. This is done by constructing a text file which contains the variables and equations of the linearised model. This file is called the TABLO Input file. The TABLO Input file is an algebraic representation of the model definition. The algebraic format of the TABLO Input file makes it relatively easy to construct as it eliminates the need for the modeller to have expertise in computer programming. Along with being relatively easy to write, the algebraic nature of the Input file means that it is also a coherent and comprehensive documentation of the model.

It is usually preferable that the initial TABLO Input file not contain any references to the data files used. This ensures that it is easy to use or modify the model for use with different data. The flexibility of GEMPACK makes it useful in developing new models from existing TABLO Input files.

After the Input file is complete the TABLO program is run; it performs the following three operations.

[5] See Pearson and Codsi 1991a,b for more detail about implementation.

1) Syntax and semantic checks — In this stage TABLO examines the Input file to check for any syntax errors (places where the format expected by TABLO has not been followed) and semantic errors (where different parts of the Input file are not logically consistent). These checks will also show where duplication has occurred or redundant information has been included.

If an error has occurred TABLO gives a brief message explaining the error. First time users should not be disheartened to find a large number of errors in their initial Input file. In many cases a large number of errors will be caused by the flow through of a single error in the Input file and can be easily rectified. The TABLO Input file must be changed to correct any errors before the implementation procedure can proceed.

If redundant information has been included, eg. a coefficient has been declared but not used, TABLO will alert you to this by warning that this information is apparently redundant. Including redundant information will not affect the running of the model but will lead to more Fortran code than necessary being generated.

2) Condensation of the model — This stage is optional. In some cases it may be desirable to reduce the size and complexity of the model by eliminating some variables. Condensation may be desirable when the model contains more detail than is necessary for the current application or because it is too costly to solve (too much time is required for execution) or exceeds the memory of the computer. This can be done by substitution, absorption or omission.

Briefly, substitution is where a variable which can be expressed as x = RHS (right hand side), is replaced wherever it occurs by the RHS of the equation. In absorption, each equation containing at least one of the variables to be absorbed is changed by removing all occurrences of all absorbed variables and replacing them with a single new composite variable. Omission as it suggests is where some variables specified in the model are left out. This would occur where the model contains more detail than necessary for a particular application. Substitution and omission are the most frequently used options in condensation, indeed Pearson and Codsi (1991a) recommend the use of omission rather than absorption.

In substitution you are replacing a variable with the components which originally determined it, ie. the RHS of the equation. This means that the substituted variable is implicitly determined within the system via it components. Therefore only endogenous variables can by substituted out of the system.

Variables omitted (or absorbed) on the other hand are eliminated (or effectively eliminated) entirely from the system of equations. There can be no determination of these variables either explicitly or implicitly. This means that only exogenous variables which are not to be shocked are suitable for omission (or absorption).

TABLO — A program within GEMPACK which accepts the model definition in algebraic form, checks for errors and automatically creates the Equations file for the model.

TABLO Input file — The text file created by the modeller containing the model definition in algebraic form.

Glossary[13]

Computable general equilibrium model — A quantitative model of an economic *system* for which numerical solutions can be calculated. These models are usually solved on a computer as they often contain many equations and draw upon large databases.

Defining the model — Deciding the purpose and economic content of the model and the base period data that will be used.

GEMPACK[14] -- General Equilibrium Modelling Package. A set of model independent software to aid in the implementation of economic models.

Implementation — The task of transcribing the model into a form that can be executed by a computer and creating the Equations file.

Johansen's method — A method for solving computable general equilibrium models. Equations in the levels of the variables of the model (quantities, prices, etc) are first replaced by linear equations whose variables are percentage changes in these levels. The model is then solved in this linearised form.

Model closure — A particular partitioning of the variables in an economic model whereby each variable is declared as either exogenous or endogenous. Also called the economic environment.

Semantic error — The error which occurs when two parts of the TABLO Input file are logically inconsistent.

Shock — A change in the value of an exogenous variable of a model that is set by the user before the model is solved. For most variables in a Johansen style solution process, the shock is a percentage change, but for some variables it is a change in magnitude (that is, in level).

Simulation — An economic experiment designed to show the effects of a given set of policy changes (shocks) introduced into the model.

Syntax error — The error which occurs when the format of the TABLO Input file is not what is expected by TABLO.

[13] See GEMPACK Document No. 11.

[14] Gempack Manager
C/- Dr K. R. Pearson
Impact Project
Monash University
Clayton Victoria 3168, Australia.

3) Automatic writing of programs — Once the TABLO Input file has been checked and contains no errors, the TABLO program automatically writes the Input file into Fortran source code. The Fortran source code is readable by those with good knowledge of this language. It is at this stage that the impact of GEMPACK can be appreciated. Instead of having to write the Fortran code for the model, GEMPACK generates this automatically from the TABLO Input file.

In the final stage of the implementation process the Fortran source code is compiled into machine code (not human readable) and linked with libraries of routines from GEMPACK. The resulting program operates on the data to produce an Equations file. When this is completed the model should be verified to ensure that it has been implemented as intended. This ends the implementation stage of GEMPACK and the model is now ready to run simulations and/or manipulate data.

2.3 SAGEM — Solution Algorithm for General Equilibrium Models

A simulation is an experiment to show the effects of a given set of policy changes or other shocks introduced into the model. For example, the modeller may wish to examine the effects of trade liberalisation on an individual economy or group of economies. In terms of comparative static methodology the simulation compares the equilibrium attained after the shocks have worked their way through the system with that which would exist without the changes.

SAGEM is the program that solves the linear set of equations that make up the model giving the results of the policy experiment being conducted.

However, before a simulation can be run the user is faced with a number of choices relating to the nature of the particular policy experiment to be conducted which will need to be faced before every simulation. These choices are: 1) what values are to be chosen for the parameters; 2) what economic closure will be adopted; 3) how many steps are to be used in the computation of the solution; and 4) for which endogenous variables are results required.[6] Each of these decisions will now be explained.

Parameter Values

The values of the parameters used in the model (eg. the elasticities of supply and demand with respect to price) are given in the database. It is the responsibility of the modeller to supply these parameters in the construction of the database.

The model closure

In all economic models under consideration here, the number of variables exceeds the number of equations. This means that the model is incapable of determining all the

[6] Dixon, Parmenter, Sutton and Vincent (1982), p. 287.

variables in the system. To overcome this problem the number of variables to be solved for is reduced by specifying the model closure. The closure is defined as the partition of the set of variables into exogenous variables (determined outside the model) and endogenous variables (those determined by the model).

For the closure to be valid the number of endogenous variables must equal the number of equations so the model is soluble. While this is a necessary condition for the closure to be economically valid it is not a sufficient condition. A closure may have the correct number of endogenous variables but still be economically meaningless. Care must be taken when specifying the closure in order to obtain sensible results.

The model closure is determined by the policy experiment being conducted and the modeller's view of how the world works. For example, one economist may believe that wages adjust quickly to clear the labour market (this does not mean that there is zero unemployment— ie that the natural rate of unemployment is zero). In this case the rate of employment will be exogenous and wages endogenous. On the other hand another economist may assume that wages do not adjust quickly and therefore that a situation of involuntary unemployment may persist in the short run. This implies that wages will be made exogenous and the rate of employment will be endogenous.

The model closure adopted will also depend on whether the short, medium or long run is being considered. Resource endowments will be exogenous in the short run because they are in fixed supply. However, in the long run they can be endogenous because the supply constraint no longer holds.

The simplest way to decide upon the closure is to endogenise those variables that are determined within the system being modelled and to exogenise those variables that are determined largely or entirely outside the system. There are three broad classes of exogenous variables. These are variables which specify the economic environment, variables which are not easily determined by economic theory, and policy variables.

For some variables such as tastes and technical change no rigorous theory about their determination exists or else the theory cannot be implemented within the model. In these cases there is little choice but to exogenise the variables.

Some government policy variables are also not rigorously determined by economic theory but are determined in a rough and ready way. Government demands in ORANI, for example, do not have any theoretical basis and so may indexed to aggregate private consumption. These variables typically appear in equations containing exogenous shift variables that allow the modeller to change the way in which the former are determined.

Number of steps to be used

Because all models developed using GEMPACK are implemented and solved via Johansen's linearisation procedure, inaccuracies may arise in the solution caused by

Example: Shares of primary factors in net factor income

```
! File declarations !

File DATIO # file containing input-output data # ;
File (new) newdata # file to contain new data # ;

! Set statements !

Set reg (reg1 - reg9) ! number of regions in SALTER model ! ;

! Coefficient declarations !

Coefficient (all,z,reg)  LTT(Z) ! total usage of labour ! ;
Coefficient (all,z,reg)  KTT(Z) ! total usage of capital ! ;
Coefficient (all,z,reg)  MTT(Z) ! total usage of land ! ;
Coefficient (all,z,reg)  DEPRL(Z) ! depreciation ! ;

! Read statements !

Read (all,z,reg) LTT(Z) from file DATIO header "AI13" ;
Read (all,z,reg) KTT(Z) from file DATIO header "AI14" ;
Read (all,z,reg) MTT(Z) from file DATIO header "AI15" ;
Read (all,z,reg) DEPRL(Z) from file DATIO header "AI20" ;

! Calculated coefficients !

Coefficient (all,z,reg) YFNT(Z)
! net factor income ! ;
Formula  (all,z,reg)  YFNT(Z)  =  LTT(Z)  +  KTT(Z)  +  MTT(Z)  -
DEPRL(Z) ;

Coefficient (all,z,reg)  SLTT(Z)
! share of labour in net factor income ! ;
Formula (all,z,reg) SLTT(Z) = LTT(Z) / YFNT(Z) ;

Coefficient (all,z,reg)  SKTT(Z)
! share of capital in net factor income ! ;
Formula (all,z,reg) SKTT(Z) = (KTT(Z) - DEPRL(Z)) / YFNT(Z) ;

Coefficient (all,z,reg)  SMTT(Z)
! share of land in net factor income ! ;
Formula (all,z,reg) SMTT(Z) = MTT(Z) / YFNT(Z) ;

! WRITE STATEMENTS !

Write SLTT to file newdata header "SLTT" ;
Write SKTT to file newdata header "SKTT" ;
Write SMTT to file newdata header "SMTT" ;
```

The output file is a Header Array file which means that it is unable to read using a text editor. The GEMPACK program, SEEHAR (See a Header Array file) allows you to convert the output to human readable form as explained in section 4.3.

nonlinearities in the model. The error associated with a linear approximation of a nonlinear model could be quite large.

To overcome this potential problem you can run a multistep simulation. This means that the shock to be implemented is broken down into a number of approximately equal parts. For example, a ten percent shock to tariff rates could be broken down into ten shocks of around one percent each. This is done by specifying a ten step simulation.

This procedure reduces the linearisation errors associated with large policy shocks on non–linear models. It does this by running a small shock, and updating the initial database and then running the next step on the updated database. This continues until all the shocks have been completed.

Before the simulation can be run the number of steps must be decided upon. This decision is made on the basis of the size of the shock(s) to be used and the modeller's expectation of the degree of nonlinearity in the system. Using this information the modeller will decide the number of steps that will be executed. The results will then be analysed to determine if the appropriate number of steps has been used.

Output of endogenous variables

The sectoral variables for which output is required are determined by the simulation experiment undertaken. The aim of the experiment will make the choice of sectoral variables obvious. The macroeconomic variables on the other hand are generally the same no matter what the experiment. Some key macro variables usually required are: real gross domestic product at factor cost (an index of input use); real net domestic product at market prices (an index of welfare); household income, savings and consumption; import and export volumes; employment and some price indexes such as net domestic product deflator and consumer price index. This is not to say that you cannot alter this list to suit your analysis.

3 Practical implementation of a model using GEMPACK

Having given a general overview of GEMPACK the next step is the practical application of the package. As this document is aimed specifically at those people who have not used GEMPACK before, this section will give a step by step outline of how to use the package from writing the TABLO Input file to creating the Equations file. Figure 2 gives an easy reference guide to the instructions necessary to use GEMPACK.

Most programs included in the GEMPACK software set can be run in two ways; interactively or in batch mode.

Running a program interactively means that you provide the necessary instructions at the terminal by responding to a series of prompts.

Batch mode (processing) means that you collect the necessary instructions for running the desired program into a single text file. Upon execution the program will run automatically, sourcing its instructions from the designated text file. The information included in the text file is exactly the same as the responses you give when running the program interactively. There are a number of examples of how to batch process GEMPACK programs and the information needed to run them (either interactively or in batch mode) below.

The advantage of using batch mode is that you need not sit at the terminal responding to prompts for the program to run. Also, the text files used in batch processing are easy to modify so you can use them for many applications and provide a record of the task.

GEMPACK runs essentially the same on several different types of computers including DOS based 80386/80486 PCs, Apple Macintosh PCs, Unix machines and VAX/VMS. However, there are a few differences between machines, namely

1) in file naming conventions,

2) the commands required to run a program, and

3) the way batch jobs are run and Stored–input files are used.

In these respects, this document applies to DOS based PCs. GEMPACK users on other machines will find that most of this document applies equally well to them, but they should be prepared for slight differences in the areas covered by 1), 2) and 3) above.

3.1 The TABLO Input file

After the model has been defined it is necessary to reproduce it in a form that your computer is able to use. This is done by transcribing the model definition into a TABLO Input file using a text editor. Commonly used text editors on DOS PCs are Blackbeard and Word 5 (when saved as a non–formatted text file); however, most text editors are adequate.

The TABLO Input file uses the TABLO Input language to specify the model. An example of this code is given in Appendix 1. The reader may also wish to consult Codsi and Pearson (1988b), who give an example of how algebraic equations would be represented in TABLO code, and Pearson and Codsi (1991b) which describes the syntax and semantics of the code. It may also be helpful to look at the TABLO Input file for an implemented model, eg. Miniature ORANI (see Codsi and Pearson 1988b), to familiarise yourself with the structure of these files.

In order to write the TABLO Input file you activate your text editor and type the appropriate code for the model. Your TABLO Input file should have the file extension TAB. Initially rather than starting the TABLO Input file from scratch it is preferable to

Appendix 1: Manipulating data using GEMPACK

The primary focus of this document has been on the development and application of computable general equilibrium models using GEMPACK. However, there is another important aspect of this package that deserves mention, namely data manipulation. The GEMPACK software is a useful tool in extracting and exploiting data.

There are two methods by which data can be manipulated in GEMPACK. The first uses TABLO Input files and the second the MODHAR program. TABLO Input files are particularly useful when conducting mathematical operations on data. On the other hand MODHAR lends itself to modification of particular elements, vectors or submatrices within data matrices or changing their dimensions. Which you use depends of the nature of the manipulation required.

This section will be confined to data manipulation using TABLO Input files. The Summary Documentation for MODHAR provides a guide to using MODHAR.

Using TABLO Input files to extract data is similar to the implementation stage of model development as can be seen in Figure 4. However, instead of preparing a model definition you will create a TABLO Input file that contains the necessary calculations to provide the required output. In order to explain data manipulation we will look at the example of finding the shares of primary factors in net factor income. The TABLO code for this manipulation is shown at the end of this appendix.

The first step in any data manipulation exercise, having declared the coefficients and files required, is to read in the appropriate data. In this instance these data will be the total usage of each primary factor (land, labour, and capital), depreciation. Net factor income is defined as total primary factor usage less depreciation.

Next you will set up the necessary calculations as shown below. These calculations use formula statements which are algebraic expressions of how the values of the coefficients are to be calculated. As you can see from our simple example the share coefficients calculated are just the usage of each primary factor divided by net factor income.

The final step of developing the TABLO Input file for manipulating data is to include write statements so that you will have access to the output. The write statements are instructions to write the values of a coefficient to a particular file or to the terminal.

Now it is just a matter of using the GEMPACK program TABLO to check the Input file for semantic and syntax errors and generate the Fortran source code. Then you will compile the Fortran code and execute the resulting program which calculates the required data and writes them to an output file. All of these steps are exactly the same as implementing a model only this time you get an output file rather than a solutions file.

You will now be prompted to respond to a series of questions about which Header Array file you want to use, what name you wish to give the output file, the number of decimal places you want and whether you want all the arrays from the old file or just a select group.

When this process is complete you will be able to access the results via the terminal or print them.

edit a pre–existing file. This not only saves time but helps new users follow the correct format.

When you have completed the TABLO Input file you will run the TABLO program to check for syntax and semantic errors, condense the model and generate the Fortran source code.

3.2 How to check your TABLO Input file

To check your TABLO Input File first transfer it onto a machine capable of running GEMPACK. As a housekeeping measure it is desirable to create your own subdirectory within the root directory of the machine being used as it will speed up the operation. Copy your Input file onto this subdirectory and remain in this directory.

The following steps should be followed to initiate the checking procedure:

1. Make sure you are in the subdirectory where your Input file is located. This is not vital but prevents output being sent to unexpected locations.

2. Invoke the GEMPACK program TABLO by entering the following command[7] ;

```
UP TABLO
```

When TABLO is first invoked you are presented with the screen as shown in Table 1.

This screen indicates the default settings for the TABLO options. These options direct TABLO to perform:

- error checks, condensation, and Fortran source code generation, and
- write all details about that run of TABLO to the Information file.

It is recommended that until you are experienced with the use of TABLO the default settings are left unchanged. To continue the checking phase press enter.

TABLO will now ask you to enter the name of your TABLO Input file. You enter the filename without the **TAB** extension as TABLO will automatically look for this file extension. TABLO will now ask for the name you wish to call the Information file. The Information file is a duplicate of the TABLO Input file with all syntax and semantic errors shown and a brief description of these errors. It also provides a list of all the variables and coefficients in the model. TABLO will ask if you wish to call the Information file the same name as the TABLO Input file with the extension **INF**. If you are happy with this name then press enter.

[7] This way of invoking TABLO applies only to DOS based PCs. On other machines, the command will be different; for example it is just "TABLO" on Unix machines.

TABLO now checks the Input file for syntax and semantic errors. At the end of this process if errors have been found then TABLO will tell you how many syntax and semantic errors have been found.

The Information file will show you where the faults are with a question mark followed by a brief explanation of the problem. You access the Information file through your text editor. All semantic and syntax errors must be corrected before TABLO will allow you to proceed to the next stage of implementation. The Information file also produces a list of variables and coefficients that are declared but not used in any equations or formula.

TABLE 1: Default settings for TABLO

```
       TABLO OPTIONS         ( — > indicates those in effect)
  First stage                Semantic check options
-------------------------------------------------------------
  F1 CHECK                   SM1 Omit semantic check
  F2 CONDENSATION            SM2 Allow duplicate names
  F3 CODE GENERATION         SM3 Omit checking coefficient
                             initialisations
                             SM4 Omit checking for warnings
                             SM5 Do not display individual
                             warnings

  Last stage                 Information file options
-------------------------------------------------------------
  L1 CHECK                   IN1 Has the same name as TABLO input
                             file
  L2 CONDENSATION            IN2 Only show lines containing
                             errors
  L3 CODE GENERATION         IN3 Omit the model summary in CHECK
                             stage
                             IN4 Omit summary of generated code
                             in CODE stage

Select an option:...               Deselect an option:...
Help for an option:...             Help on all options:...
Redisplay options:...              Finish option selection:...
Your selection >
```

4.3 Examining the Solution file

The GEMPACK program GEMPIE is used to convert a Solution file into a human readable form; GEMPIE produces a Print file. The program SLTOHA (Solution to Header Array) is also available; it converts a Solution file into Header Array format which can then be used in further analysis.

GEMPIE

Use the following command to obtain a results file with GEMPIE:

```
UP GEMPIE
```

GEMPIE will then present you with a series of options allowing you to specify the required output.

After the process is completed you will have a **EXAMPLE.PI5** file of results which can be printed or viewed from the terminal.

SLTOHA — Solution to Header Array

When obtaining the results of a simulation it is unlikely that you will require all the endogenous variables or even all the components of specific variables. SLTOHA is more flexible than GEMPIE in this respect. The output required is specified by the use of a MAP file.

A MAP file is simply a text file listing the variables and components of variables that you want. All you need to do is give the header and the components of the variable required.

Use the following command to obtain results using SLTOHA:

```
UP SLTOHA
```

SLTOHA will then prompt you for the name of the Solution file, the name of the resultant output file and whether you wish to use a MAP file and its name.

Once this procedure is completed you will have a Header Array file containing the simulation results. However, because Header Array file are binary files they are not human readable. To view a Header Array file you will need to use the program SEEHAR (See a Header Array file).

To use SEEHAR enter the following command:

```
UP SEEHAR
```

Table 6: Typical input for running a simulation

Input	Comments
`S`	Single or multistep simulation
`36`	number of steps
`Y`	use an existing Equations and BCV file
`EXAMPLE`	name of Equations file
	carriage return if the BCV file has same name
`EXAMPLE`	name of Solution file
`EXAMPLE.UPD`	name of updated database
`TMP1`	name for intermediate database
`E`	use existing closure to specify environment
`E`	closure is an Environment file
`ENVIRONMENT`	name of Environment file
`U`	use this closure
`N`[9]	do not save an LU file
`0.1`[10]	accuracy of matrix inversion
`A`	option specifying exogenous variables to be shocked
`1`	components of variables to be shocked (in this case one)
`VARIABLE`	variable(s) to be shocked
`F`	finish specifying variable(s) to be shocked
`Y`[11]	are all shocks equal to one?
`Y`[12]	are all shocks uniform?
`T`	are shocks to be entered from terminal (T) or a file (F)?
`20`	numerical value of shock
`A`	retain all endogenous variables
`**end`	end of input -- simulation will commence

9 When running many single step simulations using the same closure and Equations file it is useful to retain the LU file.

10 The value of U must be set between 0 and 1 and is typically set at 0.1.

11 Enter **N** if negative responce required.

12 Enter **N** if negative responce required.

3.3 Condensing the model and automatic code generation

When the TABLO Input file is free of errors you will be presented with the options shown in Table 2. The most important options are; [c] perform condensation, and [a] proceed to automatic code generation.

As stated earlier, condensation is an optional step. However, condensation may be necessary if the model is large or contains many superfluous variables because of the time needed to run simulations using the uncondensed model or because the computer does not have sufficient memory.

Table 2: Condensation and code generation

```
        [s] summary of the model
        [c] perform condensation
        [a] proceed to automatic code generation
        [e] exit from TABLO
```

When performing condensation interactively, you proceed as stated above, first entering the `UP TABLO` command and when there are no errors selecting option `[c]` as listed in Table 2. TABLO will then prompt you to enter manually the variables to be condensed out and whether they are to be substituted, absorbed or omitted.

The other way to carry out condensation is using a batch file containing the necessary instructions. The structure of such a file is shown in Table 3. The condensation file starts with the name of the model, ie. the TABLO Input file name without the TAB extension. This is followed by the instruction for condensation [c], the variable to be condensed out and the name of an equation where the variable occurs which will be used to eliminate all instances of the variable. This is repeated for each variable to be condensed out. It is possible to use the equation number instead of its name but this may lead to problems if more equations are included at a later date. At the end of the file the letter `e` appears twice. The first occurrence ends condensation and the second exits TABLO.

There are a number of important points to note about condensations involving substitution and the reader should consult section 8.2 in Pearson and Codsi (1987) for full detail.

One such consideration is where the variable to be substituted out is multiplied by a coefficient. Consider a substitution of variable `X` using the equation

$$a * X + b * Y = 0,$$

where `a` and `b` are coefficients. Dividing through by `a` gives the substituting expression

$$X = - 1/a * (b * Y).$$

This expression can be used to eliminate any occurrences of the variable X as long as the coefficient a is never zero. If this coefficient is zero its use in eliminating the variable X will lead to an error of attempting to divide by zero.[8]

If the coefficient of a substituted variable in the nominated equation differs from either 1 or -1, TABLO displays the coefficient and asks if the variable is suitable for substitution. If you are confident that the coefficient is always non–zero the answer will be yes. If running the condensation in batch mode place a Y (for yes) below the S in the condensation file as shown in Table 3. However, if you are not confident, you would be best advised check the coefficient or choose another variable for substitution.

Table 3: Example of a condensation file: EXAMPLE.STI

Input	Comments
SALTER	} model name
c	} condensation
s	} substitute
Y	} yes coefficient always non zero
intd	} variable to be substituted out
DOM_INT	} equation name
o	} omit
ctdgov	} variable to be omitted
a	} absorb
invt	} variable to be absorbed
e	} end condensation
e	} exit TABLO

To carry out a condensation using batch mode enter the following command:

```
UP TABLO < EXAMPLE.STI
```

TABLO will automatically check the Input file for errors and then perform the condensation (if no errors are found).

While condensation is optional, automatic code generation is not. In this stage the TABLO Input file is interpreted and Fortran source code generated as a precursor to creating the Equations file.

[8] Pearson and Codsi (1987) p. 47.

some or all of the exogenous variable(s) defined in the closure. The results of the simulation show the effects of these shock(s) on the system of equations defining the model.

As with most applications in GEMPACK you can run simulations interactively or in batch mode. The procedure to run the simulation is similar to that for generating the Equations file. However, this time you will be using the existing Equations (EQ4) and BCV files and specifying your closure from an existing one (the Environment file generated in the previous section). You will also need to input the numerical values of the shocks. Any exogenous variables that you do not shock explicitly will automatically be given a zero shock.

Table 6 shows the structure of a text file that would be used to run a simulation in batch mode along with comments about the meaning of each response. This is the same information that you will be prompted for when running a simulation interactively.

If running the simulation interactively enter the following command:

```
UP EXAMPLE
```

If using batch mode to run the simulation enter the command:

```
UP EXAMPLE <SHOCKS.STI> EXAMPLE.LOG
```

The file **EXAMPLE** is the Executable file generated by implementation, the **SHOCKS.STI** file contains the shocks and the **EXAMPLE.LOG** file is the Log file which contains output that would normally appear on the screen.

The simulation will produce a Solution file (**EXAMPLE.SL4**) and an updated database (**EXAMPLE.UPD**). The Solution file is not human readable and therefore the next step is to convert the results to a human readable form.

4 How to run simulations

Referring to Figure 2 it can be seen that after the Equations file has been created the model is ready to run simulations. Figure 3 illustrates the general procedure, the files and programs used in running a simulation.

4.1 The closure file

The first step to running a simulation is to specify the closure (the exogenous/endogenous variable split) to be used. The GEMPACK program SAGEM is used to develop the closure.

To specify the closure you can either run SAGEM interactively or in batch mode. On your first attempt to generate the closure it is a good idea to run SAGEM interactively to familiarise yourself with the necessary input.

The usual method of specifying the closure is to set all variables endogenous and then proceed to exogenise variables until you have a valid closure. Alternatively, you can set all the variables exogenous and then endogenise variables to obtain a valid closure. The advantage of these two methods is that you only need continue exogenising (endogenising) variables until the closure is valid. They also provide a succinct list of the exogenous (endogenous) variables. A third way of developing the closure is by responding to prompts. In this option SAGEM will prompt you to set each and every variable endogenous or exogenous. Responding to prompts with a large model can be time consuming, therefore the first or second method is recommended.

Some variables in your model may have a number of components. If not every component of the variable is to be exogenised (endogenised) then it is necessary to state which elements you want exogenised (endogenised).

As with the TABLO Input file, the closure must be converted to a form which is recognisable by the computer. SAGEM is used to transform the closure into a binary file. This new file is called the Environment file and is of the type `example.EN4`. If running SAGEM in batch mode to create the Environment file use the following command:

```
Up SAGEM < closure.STI
```

If running SAGEM interactively just type `UP SAGEM` and respond to the prompts.

4.2 Running the simulation

Once the closure has been specified and the Environment file generated you are ready to run the simulation. A simulation consists of specifying the shocks to be applied to

If condensation is unnecessary you will simply check the Input file for semantic and syntax errors and then select automatic code generation. You will then be presented with the options shown in Table 4. Press enter and TABLO will now transcribe the Input file into a Fortran file. It will also provide you with an Information file. These files are of the types `example.FOR` and `example.INF`.

If you have already checked and condensed the model the procedure is slightly different. When you invoke TABLO and are presented with the screen as shown in Table 1, you will select `F3 CODE GENERATION`. TABLO will then ask for the name of the model which is already checked and condensed. This procedure overrides the need to run the error checks again.

Table 4: Options for code generation

```
—> Starting code generation
    TABLO PORTABLE
    Code options ( —> indicates those in effect)

     NEQ Do no equations        DMS Do multistep code even though
     NDS Do no displays         are no updates
     NWR Do no write            SMD Code for Submatrix Data and
                                set up files
     LMC Low memory code        SPL Split code into submodels
     ACC All comment lines
     in code                    CIN Code file name as
     OCS Omit code summary      CDM Change or more default
                                maximum values in the code


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

Along with the Fortran and Information files TABLO will also generate a number of other files in this stage. These are: `example.TBR`, `example.TBT`, `example.AXS`, `example.AXT`. The `TBR` and `TBT` files include the results of checking and condensations. The `AXS` and `AXT` files are auxiliary files needed when executing the generated Fortran code. Once the Equations file has been created, the Information file and the `TB*` files can be deleted: they are no longer needed. The `AX*` files must be maintained, however, as they are needed to execute the model.

After the Fortran source code has been generated it is necessary to compile the code into machine code so that the PC can use it. The actual command to invoke compilation may vary between institutions so it is advisable to check with your GEMPACK manager for the correct command. Two commonly used commands are **`SC&L`** and **`LTG`**. To compile the Fortran code use either of these followed by the filename, eg:

```
LTG filename
```

Do not include the file extension as GEMPACK will automatically use the **`FOR`** file. The resulting file is the **`EXP`** or Executable file which is used to create an Equations file.

3.4 Creating the Equations file

The final stage in implementing the model is to generate the Equations file. In this stage the Fortran programs generated in the previous stage are executed and use the data to produce an Equations file.

The procedure used to generate the Equations file is similar to that used for running a one step simulation without getting to the stage of obtaining a Solution file.

To generate the Equations file interactively use the following command:

```
UP EXAMPLE
```

There is no need to include the extension as GEMPACK will automatically use the Executable (`EXP`) file. GEMPACK will now prompt you to respond to a number of questions as detailed in Table 5. When the process is complete you will have the Equations file which will be of the type `example.EQ4`. This task can also be accomplished using a batch file.

At the same time the Equations file is created another file called the Base Coefficient Variables or BCV file is automatically created. The BCV file contains the coefficients that are generated in the TABLO Input file. The coefficients are data generated by manipulation of the original database. Both the Equations and the BCV file are needed to run a simulation. The BCV file will be automatically accessed when the simulation is run.

Table 5: Interactively generating the Equations file

```
Do you want a single multi-step solution [s] or two or more
followed by an extrapolation [E]?
s

How many steps?
1

Do you want an existing equations or Base Coefficient Variable
(BCV) file?
N

Equations file to be created?
EXAMPLE

Type in the name of the model
(optional response)

Type in version number
1

Type in model identifier
(optional response)

Solution file to be created?
EXAMPLE

Original data file with logical name DATA
EXAMPLE.DAT

Updated version of data file with logical name DATA
EXAMPLE.UPD

How do you want to specify your closure?
From an EXISTING closure (which you may modify if you wish)?
[e]
By responding to prompts about one variable at a time?  [p]
By setting all components of all variables EXOGENOUS?  [x]
By setting all components of all variables ENDOGENOUS?  [n]
GIVE UP trying to specify a closure  [g]
g

Are you sure?
Y
```