CHAPTER 20

Solution Software for Computable General Equilibrium Modeling

Mark Horridge[*], **Alex Meeraus**[**], **Ken Pearson**[*], **Thomas F. Rutherford**[***]
[*]Centre of Policy Studies, Monash University
[**]Agricultural and Applied Economics, University of Wisconsin—Madison
[***]GAMS Development Corporation, Washington DC

Abstract

We describe the progress of computable general equilibrium (CGE) modeling software since the 1980s and contrast the main systems used today: GAMS, MPSGE and GEMPACK. The development of these general-purpose modeling systems has underpinned rapid growth in the use of CGE models, and allowed models to be shared and their results replicated. We show how a very simple model may be implemented and solved in all three systems. We note that they produce the same numerical results but have different strengths. We conclude by considering some challenges for the future.

Keywords

CGE, software, GAMS, GEMPACK, MPSGE

JEL classification codes

C63, C68, D58

20.1 INTRODUCTION

Since 1980 computable general equilibrium (CGE) modeling has evolved from a frontier technique practiced by a few specialists into a tool widely used by researchers in universities and other organizations. On the demand side, the growth has been driven by policy makers' appetite for detailed analyses of issues that affect various sectors or interest groups in different ways. Increased supply of CGE analyses has been enabled by more powerful computers and better software. The latter, software contribution, is examined in this chapter.

CGE pioneers were required to develop their own model theory, construct their own database, devise a solution strategy and hire programmers to implement that strategy in a model-specific program. Thus, new entrants to the field faced large upfront costs.

General equilibrium models require the numerical solution of large systems of equations, which in turn requires non-trivial computing power. Since the 1960s, a variety of mainframe computers offered this capacity. However, operating systems and

Handbook of CGE Modeling - Vol. 1 SET
ISSN 2211-6885, http://dx.doi.org/10.1016/B978-0-444-59568-3.00020-1

computer language (e.g. Fortran) implementations differed between manufacturers. It was not easy to "port" a program from one computer to another. This hindered the spread of CGE modeling between sites and made it hard for other modelers to replicate simulation results. Since CGE results rest upon a host of assumptions (not always documented completely or accurately), replicability — and the potential to recompute results with slightly different assumptions — underpins model credibility.

Standardization has been key to the spread of CGE modeling.[1] This includes:

(i) Consensus about many (but not all) of the model's mechanisms, e.g. the widespread use of nested constant elasticity of substitution (CES) to model substitution.

(ii) Standard data sources, such as the Global Trade Analysis Project (GTAP) world dataset that provides data for several different CGE models, and accepted file formats to distribute such data.

(iii) Widespread use of x86 PCs, usually running Microsoft Windows. Such machines are now powerful enough to run nearly all CGE models and offer a standard computer environment: the program developed in Palo Alto will run and give the same results in Bonn.

(iv) The use of a few software packages (GEMPACK, GAMS and GAMS/MPSGE[2]) to code and run most CGE models.

(v) Training in CGE methods and software at special courses, in graduate study programs or on-the-job in some large organizations, leading to a pool of CGE operatives with transferable skills.

(vi) A modest job market for CGE modelers — so that project staff can be replaced. Earlier modeling projects often withered when staff moved on.

These advances have allowed models to be widely shared and have lowered development costs for new models. Standard CGE modeling software has contributed directly, and indirectly has supported advances (v) and (vi) above. We concentrate here on the most widely used software — GAMS and GEMPACK — used by 80% or more of CGE modelers.

The remainder of this chapter is arranged as follows. Section 20.2 discusses the origins of CGE modeling software. Section 20.3 describes how general-purpose software addressed some of the problems faced by early modelers. In Section 20.4 we describe how GAMS uses a levels methodology, while GEMPACK uses a change methodology. Section 20.5 discusses features common to most CGE models. In Section 20.6 we describe in detail implementations of the same model (we call it SIMPLE) in GEMPACK, GAMS and MPSGE; we show how the results of a standard simulation are

[1] However, enough hetorodoxy and dispute remain to seed further advances.

[2] MPSGE may be considered as a high-level language used to specify CGE models, which is in principle independent of solution software. It is, however, nearly always used as a subsystem within GAMS — an experimental version that instead runs as a subsystem within GEMPACK has never gained traction.

obtained in each case; we observe that the results are the same in each case. In Section 20.7 we show how solve times change for the three systems as the number of sectors in our SIMPLE model is increased from 100 to 500. We make some suggestions about checking models in Section 20.8. In Section 20.9 we compare the features of the software packages. We end (Section 20.10) with some concluding remarks.

20.2 EARLY DAYS

By 1980 CGE was already established as a field, with models described by:

- Taylor and Black (1974) and Adelman and Robinson (1978), both associated with the World Bank.
- Dixon *et al.* (1977), building on Johansen (1960).
- Shoven and Whalley (1972) and Hudson and Jorgenson (1974).

These models were solved with special-purpose programs, usually written in Fortran. Modelers wrote the theory using algebraic notation and then had to hire a computer programmer who converted the theory of the model to Fortran programs. There were often errors and other problems since the modeler could not understand the Fortran and the programmer could not understand the model. Model changes were difficult and costly. And it was almost impossible for models to be transferred from the originator to other research groups for checking, validation or use. Indeed, often the originator was unable to reproduce results obtained as little as six months previously.

These were serious problems.

The solution recognized gradually in both US and Australia was to develop *general-purpose software* that eliminated the need to write model-specific programs in Fortran. Such software could:

- Handle a wide range of models (that is what we mean by "general-purpose").
- Accept input from modelers directly in a form which is intelligible to modelers.
- Allow modelers to make changes quickly.
- Allow modelers to concentrate on the economics (not the computing).
- Allow modelers to reliably reproduce results obtained earlier.
- Be used on a range of different computers.

Two general-purpose CGE software platforms emerged in the 1980s: GAMS in the US and GEMPACK in Australia.

GAMS (General Algebraic Modeling System) was developed starting in the mid-1970s by Alex Meeraus and Jan Bisschop working at the World Bank. It focused on the solution of large-scale, non-linear optimization problems[3] and was adapted to CGE modeling in the mid-1980s. The first GAMS-based CGE model was by Condon *et al.*

[3] The early history of GAMS is described in Kallrath (2004, pp. 20–21). The first published technical descriptions of GAMS appeared in Bisschop and Meeraus (1982) and Meeraus (1983).

(1987). Although GAMS was designed initially to handle non-CGE problems, notably optimization problems, we will only discuss the CGE capabilities of GAMS in this chapter.

Meanwhile, in Australia, the ORANI model (Dixon *et al.*, 1977, 1982) was being used widely for policy analysis. In the early 1980s ORANI was solved on the CSIRO mainframe computer. Modelers had to submit their jobs on cards and wait overnight for their simulation to run. As will be seen below (Section 20.4.2), solving ORANI required the repeated solution of large systems of linear equations. Pearson suggested that sparse solver techniques would speed up the solution of the model and delivered software to implement these techniques. Then, realizing that model implementation and changes were very difficult (because the model was coded in model-specific Fortran), he undertook to develop general-purpose software GEMPACK (General Equilibrium Model PACKage) aimed especially at solving ORANI and related models.

The first version of GEMPACK was used for teaching in 1984, and shortly after that was adopted by Australian CGE modelers. The first GEMPACK manuals were published in 1986 (see Codsi and Pearson, 1986). Early journal descriptions of GEMPACK are Pearson (1988) and Codsi and Pearson (1988). Early versions of GEMPACK only produced Johansen (one-step) solutions[4] that are not accurate, especially for large changes. The first version of GEMPACK that was able to produce accurate solutions[5] of the underlying non-linear model was Release 4.2.02 (April 1991). Hertel *et al.* (1992) compared the North American levels methods with the Australian GEMPACK-based linear methods for representing and solving general equilibrium models (see Section 20.4), and showed that the differences were those of style rather than substance since the alternative model representations and solution methods produce the same results. Early standard GEMPACK-based models producing accurate solutions were documented in Horridge, Parmenter and Pearson (1993) and Hertel *et al.* (1997).

Coming out of Rutherford's PhD thesis (1987) was a third implementation of general-purpose software, MPSGE (Mathematical Programming System for General Equilibrium analysis). This operates as a subsystem within GAMS.

By the mid-1990s, nearly all policy-important CGE models were implemented and solved using one of GAMS, MPSGE or GEMPACK. For this reason we only discuss these three software suites below.

[4] A first-order approximation of the levels equations was used to form a linear equation system, relating percentage changes in variables. The system was solved once only, producing a "Johansen" solution, so-called because it was used in Johansen (1960).

[5] GEMPACK uses the method implemented in chapter 5 of Dixon *et al.* (1982) and described in Section 20.4.2. below.

Of course it is still possible to implement and solve CGE models using other software, including spreadsheets, Mathematica, MATLAB and special-purpose code.[6] However, if you want your model to be understood by other policy modelers, it pays to use software familiar to them.

GAMS, MPSGE and GEMPACK make it easy to transfer models around the world and to reproduce model results. Most models implemented with one of these could, at least in principle, be implemented with either of the other two and yield the same results.

20.3 GENERAL-PURPOSE SOFTWARE

As mentioned above, a key motivation in designing and adopting general-purpose CGE software was to allow economists to construct and run models without being (or hiring) computer programmers or algorithm experts. So the software should make the normal tasks of specifying models, setting up and solving simulations, looking at results and writing reports as straightforward as possible. If modelers can do these things without too much effort, they are able to concentrate on the economics of their model and simulations. Thus, they are able to do qualitatively better policy analysis.

We describe below some of the main features common to today's CGE modeling systems.

20.3.1 Allow input intelligible to modelers

Each of GAMS, MPSGE and GEMPACK requires that model equations be specified in a text file, using a special syntax.

For GAMS and GEMPACK the syntax resembles standard algebraic notation used in economic text books. For example, the Leontief equation for demands for commodities by industries:

$$X_{ij} = A_{ij} Z_j, \; i \in \mathrm{COM}, j \in \mathrm{IND},$$

might be rendered for GAMS as:

```
E_X X(i,j)=A(i,j)*Z(j);
```

where subscripts *i* and *j* have been previously assigned to the sets COM and IND.

GEMPACK was originally developed for modelers used to working with the ORANI model, in which equations were written down in linearized form (relating percentage changes rather than the levels values of the variables involved). Following this

[6] The G-Cubed model (McKibbin and Wilcoxen) and the IGEM model (Jorgensen and colleagues) described in Chapters 15 and 17 of this handbook use special-purpose code.

tradition, for GEMPACK we would normally[7] write down the percentage-change (log-change) form:

```
E_x(all,i,COM)(all,j,IND) x(i,j)=a(i,j)+z(j);
```

where the lowercase symbols x, a and z are understood to be percentage changes from the initial values of X, A and Z.

Often it is clearer to explain the main features of CGE models in words, together with diagrams that show the structure of data and substitution (nesting). MPSGE builds on this way[8] of explaining models, using a concise text summary of accounting constraints and nesting structure. It assumes that most substitution behavior can be described using nested CES functions. The MPSGE representation of a model is typically rather compact.

All three systems make it fairly easy to specify (or change details of) a CGE model.

The use of the three different modeling languages is further illustrated in Section 20.6, which shows GAMS, MPSGE and GEMPACK representations of the same basic model.

20.3.2 Handle a wide range of models

While most CGE models bear a strong family resemblance, there are many differences between them. General-purpose software should be able to handle the range of CGE models used for policy analysis. Both GAMS and GEMPACK satisfy this requirement, since they allow nearly any equation to be represented. Indeed neither system contains any economics, or enforces any preconception of how a CGE model works. Either system could be used for non-CGE purposes: to compute, say, the stresses in a bridge design. The modeler is obliged to encode all economic theory used by his/her model into the text specification — making every assumption explicit. He/she is of course free to include equations that are wrong, perhaps because of typos.

By contrast, MPSGE does enforce a framework or template for models — which is, however, flexible enough to represent nearly all of the equations used in most CGE models. Within this framework, a beginner has more chance of specifying even a fairly complex model correctly. To model behavioral specifications that lie outside the basic framework, MPSGE allows exceptions to be made — at some cost to the brevity and clarity of the model code.

All three systems can handle models that are large — either because there are many equation blocks and/or because the number of sectors or regions is large. In either case such a model may be computationally demanding.

All three systems can handle models with varying levels of aggregation. For example, GTAP supplies a world model dataset with 57 sectors and more than 100 regions — too big to solve quickly. Modelers must first reduce the size of the database by combining

[7] GEMPACK also accepts equations written in levels form.

[8] GAMS/HERCULES (Drud *et al.*, 1986, 1989) took this approach even more strongly since the structure of the SAM drove model representation. However, GAMS/HERCULES is no longer available.

sectors and regions into groups, defined according to the needs of a particular simulation. The same model code can be used with any aggregation of the database – sets (i.e. lists) of regions and sectors can be read from the database at runtime.

20.3.3 Insulate modelers from the solution method

A CGE model is a large system of simultaneous non-linear equations – to solve it presents a considerable computational challenge. Hence, early CGE modelers were, perforce, amateur practitioners in the field of numerical algorithms. During the 1970s several approaches contended, including:

- Programming approaches, motivated by the wide success of the Simplex algorithm and by the idea that a CGE model can be recast as a single-optimand optimization problem. While attractive for very simple models with no distortions, this approach becomes less practical where there are multiple optimizing agents facing a range of tax and other distortions.[9]
- Scarf (1967) and related algorithms which constructed a set of equilibrium prices. These were briefly favored but discarded as slow.
- Iterative methods such as Jacobi, which required a model-specific assignment of variables to equations, and (for Gauss–Seidel) a suitable ordering of equations.
- Other methods, such as Newton–Raphson or conjugate gradient.

Today practice has converged to a common paradigm:

- The system of equations is square: the number of equations equals the number of endogenous variables.
- The solver allows a flexible assignment of variables between exogenous and endogenous groups.
- The modeler is not required to understand the internals of the solving engine – it is a "black box" to him/her.
- The solving engine does not understand the economic intent of the modeler, e.g. it does not know which variables are prices and which are quantities.
- Some or all of the model equations may be expressed as complementarities.
- An initial solution is provided which satisfies all (or nearly all) of the equations.

GEMPACK provides a single solution algorithm to satisfy these requirements. GAMS is designed to interface with a range of solvers, but most of these are optimizing[10] packages. However, the PATH solver (and its lighter-duty cousin, MILES) are direct equation solvers. Alternatively, optimizers such as MINOS or CONOPT could be used with a dummy optimand.

[9] Dixon (1975) proposed a method of constructing an artificial optimand, which corrected for distortions and multiple agents. Rutherford (1999) explores a similar idea.

[10] GAMS optimizers continue to play an important role in preparing data for CGE models, e.g. using entropy methods.

20.3.4 Windows programs to control simulations and examine data and results

Prior to the introduction of Microsoft Windows, command-line programs were used to solve CGE models. The modeler needed to remember the right operating system commands, and the names and location of files used. To examine input data and simulation results, voluminous printouts were needed. The computing experience was laborious and annoying.

Both GAMS and GEMPACK now provide Windows programs that greatly facilitate these tasks. For example, the GAMS integrated development environment (IDE) allows the user to run and manage simulations without ever seeing the command line. It remembers the right file names, and offers a data viewer to directly view two- or higher-dimensional matrices contained in binary files. GEMPACK offers a similar general-purpose IDE (WinGEM), which is supplemented by half a dozen more specialized programs — for viewing and editing code, data or results. There is a special IDE for running recursive dynamic programs.

These programs, particularly the data viewers, have greatly increased modelers' productivity. Models with many sectors, regions or periods produce a volume of results which it would be impossible to inspect in print form.

Some of the GEMPACK IDE's have another useful feature: they can identify and zip up all of the original or source files needed to reproduce a simulation. Hence, modelers can archive all the ingredients of their results, can retrieve and reproduce them fairly quickly, and can send them to others (perhaps in different countries) who can do the same.

20.3.5 Run on different computers

While today most CGE modelers work on a Windows PC, in the 1980s GAMS and GEMPACK needed to be portable across a range of computers and operating systems. Even up to the mid-1990s, the limited capacity of Windows PCs meant that larger models were sometimes solved on mainframes or UNIX computers.

For example, in the 1980s modelers were able to run ORANI only on one mainframe computer, using model-specific and operating-system-specific Fortran code.[11] Around 1989, the cost of using this computer, hosted by the Australian Industry Assistance Commission (IAC), became prohibitive. However, the ORANI solution code would only run on it and rewriting it would have taken several months. The IAC approached Pearson to see if the fledgling GEMPACK could solve ORANI on the new (and relatively inexpensive) IBM-like mainframe NAS computer they had purchased. As

[11] The story here is in no sense a criticism of the original ORANI code written by John Sutton. On the contrary, to be able to solve ORANI with the tiny (by today's, or even late 1980s, standards) memory available on computers in the late 1970s was a veritable *tour de force*. Computers improved massively during the 1980s.

GEMPACK was written using ANSI standard Fortran instructions, it was possible to port GEMPACK to the NAS and solve ORANI there within two weeks.

Portability remains in 2012, at least for the core, command-line, GAMS and GEMPACK solution programs, which can be run under, e.g. Linux or Mac environments. This is an insurance policy against future changes in computing environments.

Unfortunately the almost-indispensable Windows IDE/viewer programs are not so portable. At the same time, the introduction of 64-bit Windows means that Windows users need suffer no performance penalty. Consequently non-Windows use is mostly confined to a small band of Linux or Mac enthusiasts – and most of these will still use Windows viewer programs (either on another computer or within a Windows emulator running under their preferred operating system).

However, Windows' dominance is not quite complete. Where a CGE model is part of a larger software system – such as the "Integrated Assessment" systems that link economic and physical models – it may be necessary to run the combined system under Linux.[12] Computer clusters, which usually run Linux, might occasionally[13] be useful for, say, running Monte Carlo simulations. In each case, economists would require additional IT help to use the system.

20.4 LEVELS AND CHANGE SOLUTION METHODS

A CGE model may be represented as a system of N simultaneous non-linear equations:

$$\mathbf{F}(\mathbf{Z}) = 0.$$

The variables $\mathbf{Z}$ may be divided into N endogenous variables $\mathbf{Y}$, which are determined by the system, and the remaining variables $\mathbf{X}$ (the exogenous ones[14]). So we may write:

$$\mathbf{F}(\mathbf{Y}, \mathbf{X}) = 0.$$

As explained below, in CGE practice we assume that we know an initial solution $[\mathbf{Y}^0, \mathbf{X}^0]$:

$$\mathbf{F}(\mathbf{Y}^0, \mathbf{X}^0) = 0.$$

We wish to find another solution with different settings of some exogenous variables:

$$\mathbf{F}(\mathbf{Y}^1, \mathbf{X}^1) = 0.$$

and to report percentage differences between $\mathbf{Y}^1$ and $\mathbf{Y}^0$.

12 Essentially because some scientists and their programs will only run under Linux.

13 x86 Windows machines can contain up to 12 CPU cores, while GEMPACK is able to parallelize simulations using up to nine cores. A typical cluster offers the modeler a *share* of 100–1000 CPUs (each is which is individually much *slower* than those on a fast Windows PC), wrapped up in an unfamiliar operating system. So far, such offerings have attracted only a few.

14 These are the variables which are fixed in the GAMS code using " . FX."

Some or all equations may be defined via *complementarities*. Each complementarity equation F_i has an associated variable Y_i which is bounded between U_i and L_i. Then *just one* of these three relations will hold:

$$\begin{aligned} F_i(\mathbf{Y},\mathbf{X}) &= 0 \quad \text{and} \quad L_i \le Y_i \le U_i \\ F_i(\mathbf{Y},\mathbf{X}) &> 0 \quad \text{and} \quad Y_i = L_i \\ F_i(\mathbf{Y},\mathbf{X}) &< 0 \quad \text{and} \quad Y_i = U_i. \end{aligned}$$

Complementarities are a convenient way to represent the first-order or Kuhn–Tucker conditions which arise from optimization by agents. However, they may arise in many other contexts (e.g. quotas or stepped tax schedules).

20.4.1 Levels strategy and GAMS

The levels strategy is to search through various $\mathbf{Y}$ values to find a solution. The process is iterative; for example we might improve the current values via a Newton–Raphson step:

$$\mathbf{F}_Y\left(\mathbf{Y},\mathbf{X}^1\right)\cdot\mathbf{dy} = -\mathbf{F}\left(\mathbf{Y},\mathbf{X}^1\right),$$

where $\mathbf{F}_Y$ is the Jacobian matrix or derivative of $\mathbf{F}$ with respect to $\mathbf{Y}$ and $\mathbf{dy}$ is the suggested set of revisions to the current $\mathbf{Y}$ values. We can measure how close we are to a solution with a *merit function*, e.g. $F^{\mathrm{T}}F$ (the sum of squared errors of the equations). The process stops (i.e. we have a solution) when the merit function becomes sufficiently tiny. Near to a solution, convergence is usually very rapid (e.g. halving of error at each step).

By design GAMS does not prescribe a particular search strategy for $\mathbf{Y}$; this task is left to the variety of solvers that may be used with GAMS. Industrial-strength solvers use a range of strategies.[15] Amongst these, variants of Newton–Raphson play an important part – it may serve as an example here.

What GAMS does do is:

- Provide the modeler with a convenient interface for specifying the functions $\mathbf{F}$, setting up values for $\mathbf{X}^0$, $\mathbf{Y}^0$ and $\mathbf{X}^1$, and invoking a solver.
- Provide to the solver, on request, fresh evaluations of the functions $\mathbf{F}$ and the gradients $\mathbf{F}_Y$, using the latest estimates of $\mathbf{Y}$.

Considering the Newton–Raphson step:

$$\mathbf{F}_Y\left(\mathbf{Y},\mathbf{X}^1\right)\cdot\mathbf{dy} = -\mathbf{F}\left(\mathbf{Y},\mathbf{X}^1\right),$$

we see it requires evaluation (by GAMS) of $\mathbf{F}$ and $\mathbf{F}_Y$, and solution (by the solver) of a large linear system. All three operations are costly; GAMS (and the solvers) work to

[15] For example, the PATH solver (Ferris and Munson, undated; "PATH 4.6," GAMS documentation, http://www.gams.com/dd/docs/solvers/path.pdf) supplied with GAMS. This is able to solve models which contain both levels, equations and complementarities.

reduce costs in many ways. For example, because most variables appear in only a few equations the Jacobian matrix $\mathbf{F}_Y$ is mostly zero: GAMS' sparse storage scheme processes only the non-zero elements.

Levels solution algorithms are not indifferent to units of measurement. Suppose we declare that equations are solved if $F^T F < 10^{-6}$. The criterion will be affected by the units we choose for $\mathbf{Z}$ (whether, say, thousands or millions of dollar). Consequently:

- GAMS modelers should scale their data appropriately
- We can imagine changes in exogenous variables that are too small to notice. That is, if we have an initial solution such that $\mathbf{F}(\mathbf{Y}^0, \mathbf{X}^0)^{\mathbf{T}}\mathbf{F}(\mathbf{Y}^0, \mathbf{X}^0) < 10^{-6}$, we could find small changes $\boldsymbol{\epsilon}$ in exogenous variables such that $\mathbf{F}(\mathbf{Y}^0, \mathbf{X}^0 + \boldsymbol{\epsilon})^{\mathbf{T}}\mathbf{F}(\mathbf{Y}^0, \mathbf{X}^0 + \boldsymbol{\epsilon})$ was also $<10^{-6}$. Hence we need to ensure that shocks rise above the background level of numerical noise — we cannot directly compute the dollar change in GDP that might occur if we added $1 to national health spending. Instead we might simulate the effect of a million dollar increase — then divide simulated changes by one million.

GAMS reduces these problems by working with double precision arithmetic, which is accurate to 16 significant figures. However, compared to single precision arithmetic, double precision approximately doubles memory needs and execution time.[16]

20.4.2 Change strategy and GEMPACK

GEMPACK's solve strategy requires that we start from a solution, i.e.:

$$\mathbf{F}\left(\mathbf{Y}^0, \mathbf{X}^0\right) = 0.$$

For small changes **dy** and **dx**, in the neighborhood of a solution, we have the linear system:

$$\mathbf{F}_Y(\mathbf{Y},\ \mathbf{X}) + \mathbf{dy} + \mathbf{F}_X(\mathbf{Y}, \mathbf{X}) \cdot \mathbf{dx} = 0,$$

which we could solve to give:

$$\mathbf{dy} = -\left[\mathbf{F}_Y(\mathbf{Y}, \mathbf{X})\right]^{-1} \mathbf{F}_X(\mathbf{Y}, \mathbf{X}) \cdot \mathbf{dx}.$$

The above is a linear or first order approximation, accurate only for very small changes.[17] However, we can reduce the error as much as we like by dividing the exogenous changes $\boldsymbol{\Delta}\mathbf{X} = \mathbf{X}^1 - \mathbf{X}^0$ into N equal parts and repeatedly imposing $1/N$th of the shock to the equation above. After each step, $\mathbf{X}$ and $\mathbf{Y}$ change, so that we need to re-evaluate $\mathbf{F}_Y$ and $\mathbf{F}_X$.

[16] The time is affected because, when using double precision, only half as many numbers will fit into the fast CPU cache, and twice as many bytes must be exchanged with the main memory.

[17] Although accurate indeed for tiny changes: in this case, we *can* directly simulate a $1 increase in health spending.

This is the Euler[18] method of solving differential equations, with the following characteristics:

(i) As the number of steps N increases, we get a more accurate answer.

(ii) Unlike the Newton–Raphson method, we *never need to evaluate* **F**. However, we still need, at every step, to evaluate the derivative matrices $\mathbf{F}_Y$ and $\mathbf{F}_X$, and to solve a large linear system.

(iii) Compared to Newton–Raphson, Euler is surer to converge, but slower to reach accurate answers. We find usually that to halve the error we need to *double* the number of steps N.

However, the comfort of (i) is diminished by (ii): without evaluating **F**, how can we know how accurate our solution is or whether we need to increase N? As well, the last point (iii) seems to make Euler less attractive than Newton–Raphson, which often halves the error at *every* step.

The first, accuracy, concern is addressed as follows: we can perform three Euler simulations with, say, four, eight and 16 steps; then compare results. Then we can use these three estimates of the true answer to calculate a new estimate by Richardson extrapolation. The extrapolated estimate is typically far more accurate than any of its components; it is equivalent, say, to a 1000-step Euler calculation. As well, the extrapolation generates error bounds for each results, so we can tell if our estimate is accurate enough (if not, more steps are needed).

The second worry, that so many Euler steps may be slow, is addressed by:

- Using industrial strength linear solution routines, prepared at Harwell Labs, UK (see Duff *et al.*, 1989).
- Recasting the linearized equation system using percentage change variables. This turns out to eliminate a great deal of computation.
- Using automatic algebraic substitution of high-dimension linearized equations to enormously reduce the size of the linear system.

20.4.2.1 Simple linearized equations enable substitution

In a typical GEMPACK model, most of the change variables are expressed as percentage changes (rather than ordinary changes). Thus, the levels GAMS equation:

```
x(c,s,u)=A_ARM(c,s,u)*xcomp(c,u)*[pcomp(c,u)/p(c,s)]**SIGMA(c);
```

which describes import–domestic substitution by user u for commodity c, becomes, in percentage change form:

```
x(c,s,u)=xcomp(c,u)-SIGMA(c)*[p(c,s)-pcomp(c,u)];
```

[18] Press *et al.* (1992) think that the Euler method is useful for pedagogical purposes, but too crude for real-world use. By contrast, GEMPACK's experience with Euler has been positive and the potential for users to understand how it works is a useful bonus. Many of the tricks used by GEMPACK to improve or build on Euler are described in chapter 16 of Press *et al.* (1992, "Integration of ordinary differential equations").

The percentage change form is simpler, and uses cheaper arithmetic operations: powers are replaced by products, and products by sums. Moreover, GEMPACK code will contain commands like:

```
Backsolve x using E_x;
```

which instructs the system to drop the above equation, and simultaneously replace each occurrence of x in other equations by $\{xcomp(c,u) - SIGMA(c) * [p(c,s) - pcomp(c,u)]\}$. This can greatly reduce the number of equations in the system and is easy to do for linearized equations.

20.5 GENERAL FEATURES OF CGE MODELS

Although exhibiting great variety, CGE models tend to share various characteristics. Below we describe a common core of features from which most models depart only in a few respects. Naturally, the solution software has been adapted to cope efficiently with the most usual model features.

Like their progenitor, input-output modeling, CGE models focus on the production of commodities by industries — distinguishing from 10 to 1000 sectors. The commodities are demanded by industries and by final demanders (household, investment, government and export). Income arising from primary factor rents and from taxes and transfers is distributed to (and drives spending by) "institutions" such as households, corporations, government and foreigners.

20.5.1 Main equation groups of a model

Industry input demands are usually assumed to be proportional to output (constant-returns to scale is assumed) and sensitive to relative prices. These demand functions arise from the assumptions (i) that production (output) is a function of input quantities and (ii) that producers choose the cheapest combination of inputs that will produce a given output.

The assumptions impose some (triad) restrictions on the form of input demands — but leave open a wide range of substitution possibilities. To tie these down, it is usual to impose additional, separability, assumptions. For example, if an industry has multiple inputs and multiple outputs, we may stipulate an activity index Z such that:

$$F(\text{inputs}) = Z = G(\text{outputs}).$$

That is, a sector will choose the cost-minimizing mix of inputs to produce Z and, given Z, will choose (subject to G) a revenue-maximizing output mix. The implication is that input mix is independent of output mix or that large blocks of cross-price elasticities are zero, greatly simplifying computation and elasticity estimation. Very often the F function will also be separable via:

$$F(\text{inputs}) = H(I(\text{primary factors}), J(\text{material inputs}))$$

Similarly an industry's choice of how much, say, flour, to use, will be modeled independently of the choice whether to use domestic or imported flour. Household, government and investment demands are modeled in a similar way.

Thus complicated demand systems are built up from a series of subaggregator functions, called *nests*. Usually the functional forms used are of the CES type or are generalizations of CES. Diagrams of the nesting structure is often described by a diagram, like Figure 20.1 in Section 20.6.

The remaining core equations of the typical CGE model are fairly simple, consisting of:

- Market-clearing equations that add up total demands for each good and, for domestic goods, equate demand to domestic output.
- Price equations that relate tax-inclusive user prices to output or c.i.f. (cost, insurance and freight) prices.
- Income equations that add up total revenue accruing to each agent.

Usually there will, in addition, be equations defining common macro aggregates such as GDP or absorption, and model-specific equations defining, say, wage-indexation assumptions or rules for investment and capital accumulation.

20.5.1.1 Block equations: Industry detail captured in matrices of numbers

Usually one (or a small number) of behavioral specifications is applied to all sectors, households or regions. That is, the same functional form is used for all, say, household demand equations. Differences between goods or household types are captured in good- or household-specific numbers held on or deducible from the model database. The equations of the model are thus vector or matrix equations — expressing in a few lines of

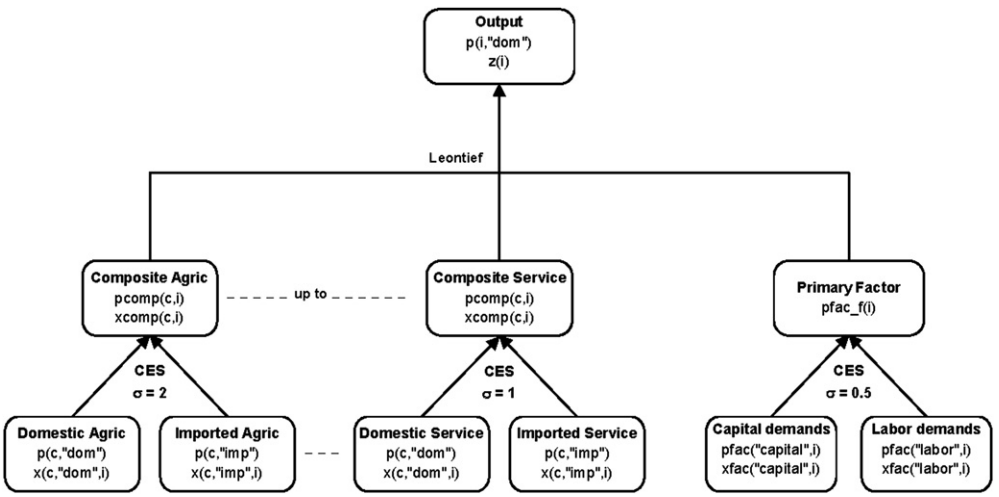

Figure 20.1 Structure of input demands for industry *i*.

code, for example, demands by every household type for any of the commodities. This scheme enables a huge number of equations to be represented compactly.

20.5.1.2 Data: Social accounting matrix (SAM) plus elasticities

The typical CGE model database consists of:

- *Matrices of transaction values*, showing, for example, the value of coal used by the iron industry. Usually the database is presented as an input-output table or as a SAM. In either case, it covers the whole economy of a country (or even the whole world), and distinguishes a number of sectors, commodities, primary factors and perhaps regions or types of household.
- *Elasticities*: dimensionless parameters that capture behavioral response. These include the CES substitution parameters that underlie nested production and demand structures, and export demand or household expenditure elasticities.

The SAM structure, which arranges all flows into a single table according to the recipient (row) and payer (column), is a useful conceptual and pedagogical tool. In practice it can become inconvenient for large or detailed models since the SAM becomes too large or sparse to view easily. Also, the need to distinguish all flows through only two subscripts is constricting. For example, if we distinguish goods by sector and source, and households by income and region, the natural way to record household demands is:

$$V_{(i,s,d,r)} \quad \text{spending on good } i \text{ from source } s \text{ by income decile } d \text{ in region } r,$$

i.e. to use four subscripts. In a two-subscript SAM schema, this appears as:

$$V_{(g,h)} \quad \text{spending on good } g \text{ by household type } h,$$

where subscript g runs over a compound set containing all combinations of good and source, and subscript h similarly combines all deciles and regions. Often therefore CGE flows data that could in principle be combined into an enormous, mostly-zero, SAM, are instead stored as a series of multidimensional matrices representing the non-zero blocks of a hypothetical SAM.

Usually these data are stored in a binary file of proprietary format.[19] While both GAMS and GEMPACK are able to read data in text or spreadsheet formats, binary storage is more compact and faster to process, especially for large models.

20.5.2 Database is consistent with initial truth of the model equations

Typically the flows database represents a single year and is assumed to be consistent with the model equations. This point deserves elaboration.

[19] For example, GEMPACK's .HAR files and GAMS' .GDX files. Tools are available to easily convert between these two or into other common formats (e.g. text).

Suppose that intermediate input demand by sector j for good i (X_{ij}) is given by the Leontief function:

$$X_{ij} = A_{ij}Z_j,$$

where Z_j is output of sector j and A_{ij} is a technological parameter. None of these variables appear in the model database, which only supplies:

$$V_j = P_jZ_j \quad \text{the value of output of sector } j$$

$$V_{ij} = P_iX_{ij} \quad \text{the value of good } i \text{ used by sector } j.$$

To obtain initial values for the variables we need to make assumptions − this is called *calibration*. For example, we might assume that both P_j and P_i were 1. Given the flow values, we could then deduce the quantities X_{ij} and Z_j, and thence calculate A_{ij}. Clearly such a procedure guarantees (or assumes) that the demand equation is initially true.

The assumption that initial prices are 1 amounts to no more than an arbitrary choice of physical units for quantities like "light industrial" output or demand for "other services" that have no natural unit of measurement. As a result, it is meaningless to report simulated levels values of prices and quantities, except in relation to their initial values. So, universally, CGE model results are presented in terms of *percentage changes* from the initial levels values.

One might ask, would results for a given simulation be affected if we had assumed that $P_j = 2$ and $P_i = 3$ (instead of both 1). Clearly, both initial and final levels results would be affected, but the percentage change results would be the same − otherwise we would reject the model or calibration process.

Like physical units, monetary units are irrelevant to percentage change results. We could halve all initial data flow values (converting from dollars to pounds) without altering the percentage change results from any simulation.

The database also stores various elasticities which affect behavior. By definition, these are unitless − so initial levels values are important. They are not "free" in calibration. Consider the CES equation:

$$K/L = A \cdot (P_L/P_K)^{\sigma}.$$

Assuming prices to be 1, and given initial flow values of capital and labor demand, we can deduce K and L and so deduce A. The value of σ must be supplied.[20]

[20] The alert reader will notice that, with initial price ratio 1, a shock to σ will have no effect; yet with non-uniform initial prices, a shock to σ (holding A fixed) *would* have an effect − contradicting the claim that assumed initial prices do not affect percentage change results. The short retort might be that this is why mature CGE modelers never shock elasticities. A longer answer might explore the way in which different ways of writing the CES functions, which are generally agreed to be equivalent, in fact act differently if σ is considered to be a variable.

It follows from this discussion that the *whole* useful content of a CGE model could be contained in a matrix function F:

$$[p, q] = F(D, E, t, a),$$

p and q represent finite[21] percentage changes in endogenous model variables, D represents various matrices of shares or ratios derived from the SAM or input-output database, E is the set of elasticities, and t and a are percentage changes in exogenous variables, such as tax rates or technological coefficients.[22] Both the GEMPACK and the MPSGE ways to represent a CGE model draw on this insight.

The format of matrix function F above minimizes computation cost in the sense we only need to compute and store D and E (the initial database) and p, q, t and a (the results we need to report). That is, an average of three values per cell in the original input-output table (p, q, D).[23] A full levels-based approach is more expensive, since we need to store and compute the:

- Original flows values $V0$, prices $P0$, quantities $X0$ and technological coefficients $A0$.
- Post-solution values of P and X (these were initially set to $P0$ and $X0$).
- Percentage changes p and q calculated as $p = 100(P/P0 - 1)$ and $q = 100(Q/Q0 - 1)$.

That is, an average of eight values per cell in the original input-output table. The load is more than doubled,[24] just to support the fictional construct of levels values.

20.5.3 General features of model code

For both of the two main CGE modeling systems, GAMS and GEMPACK, plain text files are used to specify the CGE model. In GAMS the same text file could also specify several individual simulations; GEMPACK requires that each simulation is specified by its own file, separate from that specifying the model.

Both GAMS and GEMPACK provide their own computer languages, to describe model and simulations. As in most computer languages, entities such as sets or variables must be described or "declared" before they are used. This imposes an overall structure on model specification files:

- The first step is to define, either by direct enumeration or by reading from a file, the size and contents of various key sets — of sectors, regions, and so on.
- Next, data arrays are declared and their values are read from file.

21 The percentage changes are not infinitesimal and F is non-linear.

22 The conclusion depends only on the need for model results to be independent of arbitrary choice of units, and requires no controversial neoclassical assumptions. For example, money illusion or increasing returns to scale could be incorporated, if desired, via additional elasticities in the database.

23 We assume that only a tiny proportion of the exogenous variables t and a are shocked. The remainder are by definition zero.

24 That is, 8/3. In practice this ratio is often reduced to 5/3 by such devices as assuming that $V0 = X0$ and reporting only a tiny subset of percentage change results. The first device has the disadvantage that we must apply and follow some special convention about initial prices, which might be complicated if prices are user-specific. The second device makes it harder to understand, in detail, what the model is doing or if it is working as desired.

- More data arrays are declared and given values that are derived from file data.
- Next, in GAMS code, units of measurement are assigned (usually by setting prices) and numerous multiplicative constants are computed. The logarithmic form of percentage-change equations obviates the need for this step in GEMPACK.
- Next the model equations are given, in levels or (for GEMPACK) in percentage-change form.
- The closure or choice of exogenous variables is specified.
- A GAMS model usually includes a no-shock simulation as a check; it should reproduce the benchmark data.
- An exogenous variable is changed and a new solution computed.
- In GAMS, additional code is needed to derive percentage-change results from old and new levels values.

20.6 THREE REPRESENTATIONS OF A SIMPLE MODEL

We begin this section by showing the code for a fairly standard model that we call SIMPLE. We describe the model using data and nest diagrams, and then give details of implementations[25] in GEMPACK, GAMS and MPSGE.

Table 20.1 shows the model database. Five competitive single-product industries (Agric, …, Service) produce corresponding goods which are used by industries and final demanders. All non-export users also use imported goods, and industries also use Capital and Labor. There are no taxes in the model.

Demands are structured by a series of nests. For industries, the production/demand structure is shown in Figure 20.1. Each industry combines domestic and imported equivalents into commodity-composites via a CES aggregator. The elasticity of substitution varies by commodity and is shown in the last row of the table above. Import/domestic shares vary by good and by user. Industries demand commodity-composites in proportion to output. Similarly, each industry combines capital and labor using a CES with value 0.5 to produce a primary-factor-composite, which is also demanded in proportion to industry output.

For the non-export final demanders (Hou, Gov and Inv), Figure 20.2 shows a similar structure except labor and capital are not used, and the topmost nest is Cobb-Douglas rather than Leontief. Exports face foreign demand curves with a constant elasticity of −5.

To close the model, we will assume that industry capital stocks are fixed, while labor is mobile between sectors, but fixed in aggregate. Aggregate nominal values of the Hou, Gov and Inv demands are fixed proportions of nominal GDP. Other macro closures would be possible.

[25] All three model specification files and their common database are downloadable: item TPMH0103 at http://www.monash.edu.au/policy/archivep.htm.

Table 20.1 Database for example model

	Sectors					Final Demands				
	Agric	**Mining**	**Manuf**	**Housing**	**Service**	**Hou**	**Gov**	**Inv**	**Exp**	**Total**
Dom										
Agric	8	1	45	0	0	0	1	0	70	125
Mining	3	2	33	0	1	4	1	0	92	136
Manuf	9	25	12	4	15	50	27	98	6	246
Housing	1	0	0	0	2	54	3	0	0	60
Service	11	12	11	9	5	30	31	32	22	163
Imp										
Agric	1	0	5	0	0	0	0	0	—	6
Mining	1	1	10	0	0	1	0	23	—	36
Manuf	4	4	18	0	15	30	6	56	—	133
Housing	0	0	0	0	0	0	0	0	—	0
Service	0	3	3	0	2	1	6	9	—	24
Capital	43	41	50	47	52	—	—	—	—	233
Labor	44	47	59	0	71	—	—	—	—	221
Total	125	136	246	60	163	170	75	218	190	—
Sigma (*dom*/*imp*)	*2*	*1*	*2*	*0*	*1*	—	—	—	—	—

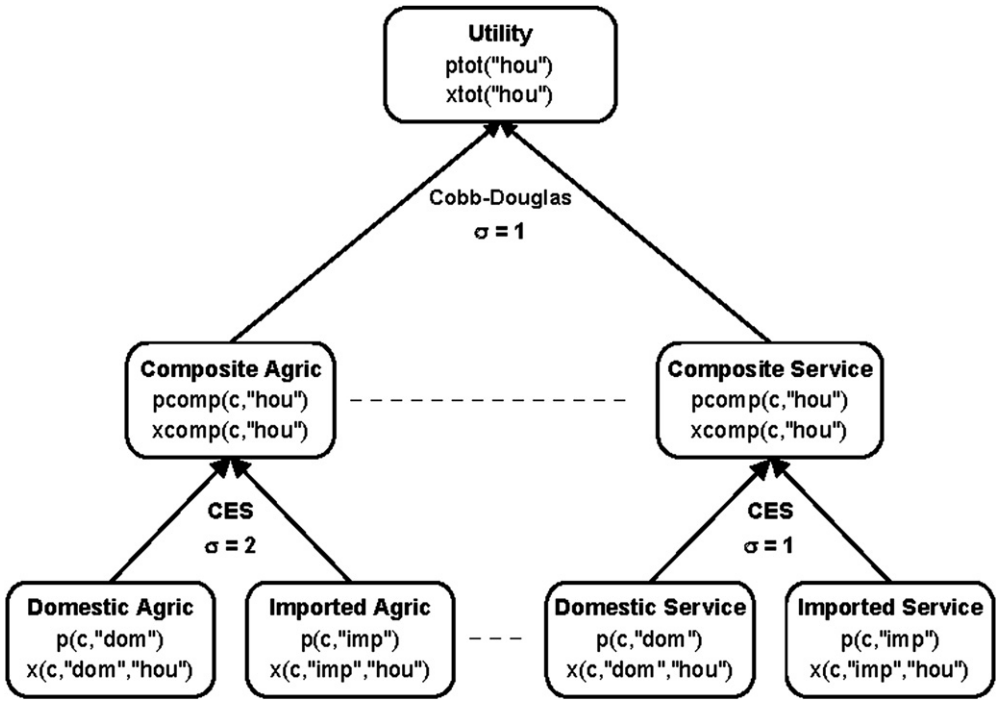

Figure 20.2 Structure of non-export final demand.

To be sure that our database is consistent with model equations (i.e. that we can calibrate a solution — see Section 20.5.2), we need to check only two things:

- All the numbers are greater than or equal to 0.
- The first five row totals, showing the value of sales of each domestic good, equal the first five column totals, showing the total cost of each industry.

20.6.1 SIMPLE model in GEMPACK

Table 20.2 shows how the model is specified using GEMPACK's TABLO language. The file starts by defining various sets. The list of sectors is read from a data file — so the same code could be used for a model with more or different sectors.

Next, the three data arrays (*USE*, *FACTOR* and *SIGMA*) are defined and their values read from file. The notation:

```
(all,f,FAC)(all,i,IND)FACTOR(f,i)
```

indicates that *FACTOR* is a matrix with two rows (Capital and Labor, the elements of set FAC) and five columns (the five sectors that form set IND).

Then, several other coefficients are defined which are all simple addups of *USE* and *FACTOR*. The `Assertion` statement is a safety check — the solver will terminate with an error if DIFF is not tiny.

As for most GEMPACK models, this simple model is specified in percentage change form.[26] The model variables are listed next. By default, these are defined as percentage changes from the initial solution, except for the first, *delB*, which is an ordinary change.

As explained in Section 20.5.2, the percentage change formulation allows us to express the model, and to compute percentage changes in prices and quantities, without ever specifying initial values for prices and quantities (we only need initial flows). This enables brevity in model specification, and reduced computation time.

The equations that follow are all linear in percentage change variables, with coefficients that are database flows, addups of these flows or elasticities. The first equation block:

```
E_pcomp # Price: dom/imp composites #(all,c,COM)(all,u,LOCUSR)
 ID01[USE_S(c,u)]*pcomp(c,u)=sum{s,SRC,USE(c,s,u)*p(c,s)};
```

is a group of equations that define, for each good and local user, the first-order change in the CES price index of the composite good which combines domestic and imported versions. It states that *pcomp*(*c*,*u*) is a cost-share-weighted average of the percentage changes in domestic and imported prices (Roy's Identity). The built-in function **ID01** takes care of the case when use of the composite, *USE_S*(*c*,*u*), is zero: **ID01**(*x*) = *x* for all non-zero *x*, while **ID01[**0**]** = 1. Thus, in the degenerate case we have *pcomp*(*c*,*u*) = 0. The name of the equation block, *E_pcomp*, is chosen to suggest that these equations can determine values of the variable *pcomp*.

[26] GEMPACK also allows you to specify the model in the levels — this would lead to model code which rather resembled the GAMS version.

Table 20.2 TABLO input file for example GEMPACK model

```
File INFILE # base data file #;
Set
 COM # Commodities # read elements from file INFILE header "SEC";
 IND # Industries #  read elements from file INFILE header "SEC";
 FIN # Final users # (Hou,Gov,Inv,Exp);
 LOCFIN # Local final users # (Hou,Gov,Inv);
 USR # All users # = IND + FIN;
 LOCUSR # Local users # = IND + LOCFIN;
 SRC (dom,imp);
 FAC (Capital,Labor);
 SEC = COM intersect IND;
Subset LOCUSR is subset of USR;
Coefficient ! data on file !
(all,c,COM)(all,s,SRC)(all,u,USR) USE(c,s,u) # Value of inputs #;
(all,f,FAC)(all,i,IND)           FACTOR(f,i) # Primary factor costs #;
(parameter)(all,c,COM)              SIGMA(c) # CES dom-imp elasticity #;
Read
 USE    from file INFILE header "USE";
 FACTOR from file INFILE header "FACT";
 SIGMA  from file INFILE header "SIG";
Coefficient ! calculated !
(all,c,COM)(all,u,USR) USE_S(c,u) # Value of import composites #;
(all,i,IND)             VALADD(i) # Factor costs #;
(all,u,USR)              COSTS(u) # Costs #;
(all,c,COM)(all,s,SRC) SALES(c,s) # Sales #;
(all,q,SEC)               DIFF(q) # Costs-Sales #;
                             VGDP # GDP #;
Formula
(all,c,COM)(all,u,USR) USE_S(c,u) = sum{s,SRC, USE(c,s,u)};
(all,i,IND)             VALADD(i) = sum{f,FAC,FACTOR(f,i)};
(all,u,USR)              COSTS(u) = sum{c,COM, USE_S(c,u)};
(all,i,IND)              COSTS(i) = COSTS(i) + VALADD(i);
(all,c,COM)(all,s,SRC) SALES(c,s) = sum{u,USR, USE(c,s,u)};
(all,q,SEC)               DIFF(q) = COSTS(q) - SALES(q,"dom");
                             VGDP = sum{i,IND,VALADD(i)};
Assertion # Check data balance # (all,q,SEC) ABS[DIFF(q)]<0.001;
Variable
            (change) delB # (Nominal balance of trade)/{nominal GDP} #;
                     xgdp # Real GDP #;
                  wgdpinc # Nominal GDP from income side #;
                  wgdpexp # Nominal GDP from expenditure side #;
                      phi # Exchange rate (local$/foreign$) #;
      (all,c,COM) xexp(c) # Export quantities #;
     (all,c,COM) fqexp(c) # Right shift in export demand curve #;
         (all,i,IND) z(i) # Industry outputs #;
      (all,c,COM) pfimp(c) # Imp goods prices, foreign $ #;
```

(*Continued*)

Table 20.2 TABLO input file for example GEMPACK model—cont'd

```
          (all,l,LOCFIN) wtot(l) # Nominal expenditure by local final users #;
          (all,l,LOCFIN) xtot(l) # Real expenditure by local final users #;
          (all,l,LOCFIN) ptot(l) # Price indices for local final users #;
           (all,f,FAC) xfac_i(f) # Total value-weighted factor use #;
           (all,f,FAC) ffac_i(f) # Factor wage shift #;
(all,c,COM)(all,s,SRC)    p(c,s) # Factor prices #;
(all,c,COM)(all,s,SRC) xdem(c,s) # Total demand for goods #;
(all,f,FAC)(all,i,IND) pfac(f,i) # Factor prices #;
(all,i,IND)            pfac_f(i) # Factor composite prices #;
(all,f,FAC)(all,i,IND) xfac(f,i) # Factor use #;
(all,f,FAC)(all,i,IND) afac(f,i) # Factor-using technical change #;
(all,f,FAC)(all,i,IND) ffac(f,i) # Factor wage shift #;
(all,c,COM)(all,l,LOCUSR) pcomp(c,l) # Price dom/imp composites#;
(all,c,COM)(all,l,LOCUSR) xcomp(c,l) # Quantity dom/imp composites#;
(all,c,COM)(all,s,SRC)(all,l,LOCUSR) x(c,s,l) # Commodity demands #;

Equation
! Dom/imp CES BLOCK !
E_pcomp # Price: dom/imp composites # (all,c,COM)(all,u,LOCUSR)
 IDO1[USE_S(c,u)]*pcomp(c,u)=sum{s,SRC,USE(c,s,u)*p(c,s)};
E_x # Quant: final dom/imp composites# (all,c,COM)(all,s,SRC)(all,l,LOCUSR)
 x(c,s,l) = xcomp(c,l) - SIGMA(c)*[p(c,s)-pcomp(c,l)];
! Industry demands !
E_pA # Industry cost indices # (all,i,SEC)
 IDO1[COSTS(i)]*p(i,"dom")=sum{c,COM,USE_S(c,i)*pcomp(c,i)}+VALADD(i)*pfac_f(i);
E_pfac_f # Factor composite prices # (all,i,IND)
 IDO1[VALADD(i)]*pfac_f(i) = sum{f,FAC, FACTOR(f,i)*[pfac(f,i)+afac(f,i)]};
E_xfac # Factor demands # (all,f,FAC)(all,i,IND)
 xfac(f,i) = z(i) + afac(f,i) - 0.5*[pfac(f,i)+afac(f,i)-pfac_f(i)];
E_xcompA # Local final demands (Cobb—Douglas) # (all,c,COM)(all,i,IND)
 xcomp(c,i) = z(i);
! Final demanders !
E_xcompB # Local final demands (Cobb—Douglas) # (all,c,COM)(all,l,LOCFIN)
 xcomp(c,l) + pcomp(c,l) = wtot(l);
E_wtot # Absorption # (all,l,LOCFIN) wtot(l) = wgdpinc;
E_ptot # Price indices for local final users # (all,l,LOCFIN)
 IDO1[COSTS(l)]*ptot(l) = sum{c,COM, USE_S(c,l)*pcomp(c,l)};
E_xtot # Real expenditure by local final users # (all,l,LOCFIN)
 xtot(l) + ptot(l) = wtot(l);
E_xexp # Export demands # (all,c,COM)
 xexp(c) = fqexp(c) -5.0*[p(c,"dom")-phi];
! Total demands and market clearing !
E_ffac_i # Total (value-weighted) factor use # (all,f,FAC)
 sum{i,IND,FACTOR(f,i)*[xfac_i(f) - xfac(f,i)]} = 0;
E_xdem  # Total demand for goods # (all,c,COM)(all,s,SRC)
 IDO1[SALES(c,s)]*xdem(c,s) =
```

Table 20.2 TABLO input file for example GEMPACK model—cont'd

```
  sum{l,LOCUSR,USE(c,s,l)*x(c,s,l)} + USE(c,s,"exp")*xexp(c);
E_z    # Market clearing # (all,i,SEC) z(i) = xdem(i,"dom");
! Miscellaneous !
E_pB   # Import prices # (all,i,SEC) p(i,"imp") = pfimp(i) + phi;
E_pfac  # Factor mobility/remuneration # (all,f,FAC)(all,i,IND)
 pfac(f,i) = ptot("hou") + ffac(f,i) + ffac_i(f);
E_wgdpinc # Nominal GDP from income side #
 VGDP*wgdpinc = sum{i,IND, sum{f,FAC, FACTOR(f,i)*[pfac(f,i)+xfac(f,i)]}};
E_xgdp  # Real GDP from income side #
 VGDP*xgdp = sum{i,IND, sum{f,FAC, FACTOR(f,i)*[xfac(f,i)-afac(f,i)]}};
E_wgdpexp # Nominal GDP from expenditure side #
 VGDP*wgdpexp= sum{c,COM, sum{l,LOCFIN, USE_S(c,l)*wtot(l)}
 + USE(c,"dom","exp")*[p(c,"dom")+xexp(c)]
 - SALES(c,"imp")*[p(c,"imp")+xdem(c,"imp")]};
E_delB  # (balance of trade)/GDP #
 100*VGDP*delB = sum{c,COM,USE(c,"dom","exp")*[p(c,"dom")+xexp(c)-wgdpinc]}
               - sum{c,COM,SALES(c,"imp")*[p(c,"imp")+xdem(c,"imp")-wgdpinc]};
Update
(all,i,IND)(all,f,FAC) FACTOR(f,i)=pfac(f,i)*xfac(f,i);
(all,c,COM)(all,s,SRC)(all,l,LOCUSR) USE(c,s,l) = p(c,s)*x(c,s,l);
(all,c,COM) USE(c,"dom","exp") = p(c,"dom")*xexp(c);
Backsolve x using E_x; pcomp using E_pcomp;
```

The next equation block:

```
E_x # Quant: final dom/imp composites# (all,c,COM)(all,s,SRC)(all,l,LOCUSR)
 x(c,s,l) = xcomp(c,l) - SIGMA(c)*[p(c,s)-pcomp(c,l)];
```

specifies pairs (for s = "dom" or "imp") of demand equations for each good and non-export user. We could subtract the imported from the domestic equation to demonstrate the CES property:

```
(all,c,COM)(all,l,LOCUSR)
 x(c,"dom",l) - x(c,"imp",l) = SIGMA(c)*[p(c,"imp")-p(c,"dom")];
```

Many of the remaining equations follow similar patterns, or are simple addups of nominal values. As a check on the model's accounting relations, we have included two variables and equations for nominal GDP: we expect, from Walras' law, that both variables will end up with the same value.

The equation for real GDP: [27]

```
E_xgdp  # Real GDP from income side #
 VGDP*xgdp = sum{i,IND, sum{f,FAC, FACTOR(f,i)*[xfac(f,i)-afac(f,i)]}};
```

is a Divisia index, not translated from any levels equation.

[27] A more general version of this equation is derived in Section 19.3.3 of Chapter 19 by Dixon and Rimmer in this Handbook.

The rule for primary factor wages:

```
E_pfac  # Factor mobility/remuneration #(all,i,IND)(all,f,FAC)
pfac(f,i)=ptot("hou")+ffac(f,i)+ffac_i(f);
```

is of a distinctively Dixonian[28] form. At runtime the "*f*" variables ("shifters") will be set exogenous or endogenous to enforce different factor market closures. For example, if for capital we set *ffac_i*("Capital") exogenous and allowed the *ffac*(*i*,"Capital"), which appear nowhere else, to float endogenously, the equation would be otiose, merely determining changes in *ffac*. We could fix industry capital stocks, *xfac*(*i*,"Capital"), to determine the capital rents. For a slack labor market we could hold both *ffac*("Labor," *i*) and *ffac_i*("Labor") fixed, so indexing wages to the CPI. Or we could fix both total labor demand, *xfac_i*("Labor"), and the *ffac*("Labor," *i*) whilst allowing *ffac_i*("Labor") to float endogenously. The effect would be that industry wages would move together, to equate demand and supply for sectorally mobile labor.

At the end of SIMPLE.TAB are Update instructions which tell GEMPACK how values on the data file are related to model variables. For example, the statement:

```
Update (all,i,IND)(all,f,FAC) FACTOR(f,i)=pfac(f,i)*xfac(f,i);
```

indicates that the coefficient *FACTOR* is a product (in the levels) of prices PFAC and quantities XFAC. GEMPACK uses this information to update data flows after each step in the calculation. *FACTOR* is updated to become:

$$FACTOR(f,i) * [1 + 0.01 * pfac(f,i) + 0.01 * xfac(f,i)].$$

Finally, come backsolve statements such as:

```
Backsolve x using E_x;
```

This tells GEMPACK to perform algebraic substitution on the linear system, everywhere replacing $x(c,s,u)$ by $\{xcomp(c,u) - SIGMA(c) * [p(c,s) - pcomp(c,u)]\}$. This enormously reduces the size of the equation system. The values of x are recovered and are available on the solution file. However, no element of x can be exogenous.

20.6.2 Example simulation

In GEMPACK, each simulation is specified using a separate file, suffixed ".CMF." Such a file is shown in Table 20.3. It starts by identifying the model, SIMPLE, described in the file SIMPLE.TAB (listed in Table 20.2). The name of the input file (simdata.har) is given. A file of identical format will be produced containing postsimulation (updated) values of all the data — this will be named after the .CMF file.

On each line, any text following an exclamation mark is treated as a comment.

[28] The device of using closure changes to enforce alternate behavioral rules was introduced in ORANI by Dixon *et al.* (1977, 1982) and continues to be heavily used in successor models.

Table 20.3 GEMPACK Command file for sample simulation

```
auxiliary files = simple;
log file = yes;
File INFILE = simdata.har; !
Updated File INFILE = <cmf>.UPD; ! will be named after CMF
method = GRAGG ; ! Solution method
steps = 4 6 8;
! Automatic closure generated by TABmate Tools...Closure command
!     Variable   Size
Exogenous     afac ; ! FAC*IND  Factor-using technical change
Exogenous     ffac ; ! FAC*IND  Factor wage shift
Exogenous    fqexp ; ! COM      Right shift in export demand curve
Exogenous     pfimp ; ! COM      Imp goods prices, foreign $
Exogenous      phi ; ! 1        Exchange rate (local$/foreign$) [Numeraire]
Exogenous  xfac_i ; ! FAC      Total value-weighted factor use
Rest endogenous;    ! end of TABmate automatic closure
!  new exogenous      old exogenous
swap xfac("capital",IND) = ffac("capital",IND); ! fix industry capital stocks
swap ffac_i("capital") = xfac_i("capital");  ! unfix total capital stock
Verbal Description = Improvement in Service labor productivity;
shock afac("Labor","Service") = -10;
```

The next lines cause three Gragg[29] simulations to be run, with four, six and eight steps. The final answers are produced by extrapolating from results of these three simulations.

Next comes the list of exogenous variables or closure. GEMPACK users customarily begin from a well-known or standard closure; then, using "swap" statements, define differences from that known closure. In this case the reference closure is produced automatically by the software — it consists of all those variables that do not have an equation named after them. In this reference closure both labor and capital are mobile between sectors and fixed in aggregate; however, we wish to fix industry capital stocks.

The statement:

```
swap xfac("capital",IND) = ffac("capital",IND);
```

causes the capital parts of the *xfac* variable (currently endogenous) to be set exogenous, while the capital parts of the *ffac* variable (currently exogenous) will become exogenous. This mechanism ensures that the number of endogenous variables (which *must* equal the number of equations) does not change. Having fixed the parts, we must endogenize the whole — this is done by a second swap statement.

[29] The Gragg (or modified midpoint) method is a variation of the Euler method of solving equations (see Section 20.4.2).

Last comes a short description of the simulation, followed by the imposed change in shocked variables. In this case we reduce by 10% the amount of labor needed to produce a unit of services.

Results from the simulation are shown (as percentage changes from base) in Table 20.4. First note that with aggregate nominal household, government and investment demands following nominal GDP, the residual balance of trade must also be a constant fraction of GDP. Service becomes cheaper and that industry expands. Labor is released to other sectors, wages fall, and capital increases its share of national income (recall, the labor/capital CES is 0.5). Output increases in most sectors, especially Manuf and Service, while Agric and Mining, which are more export oriented, expand less. Housing output cannot rise because its capital stock is fixed and it uses no labor.

The same results will be generated by GAMS and MPSGE with some trivial differences. For example, the GEMPACK results shows changes in the quantity of housing exports — even though Table 20.1 shows no housing exports. GEMPACK users are accustomed to generating such results, which may be safely ignored and never affect the percentage changes in genuine, non-zero, flows.

20.6.3 SIMPLE model in GAMS

Table 20.5 shows how the SIMPLE model may be implemented[30] in GAMS. In this case a single .GMS text file replaces the .TAB and .CMF files used by GEMPACK. As far as possible the same notation has been used as in the GEMPACK implementation. The chief difference is that although results are again reported as percentage changes, the model is specified in the levels. This means that for all quantities and prices, separate symbols must be defined to represent initial values, final values and percentage changes. In addition, the levels equations require a number of multiplicative constants (which drop out in percentage change form). The need to declare and assign values to all these parameters and variables takes a bit more space than needed in GEMPACK.

Section 1 of the GMS file reads in sets and data from the .GDX file, and defines some additional sets. Then, zeroes in the factor payments data are replaced by tiny numbers, while the degenerate case where a CES = 1 is avoided. This simplifies subsequent code, although other approaches are possible. Next some useful add-ups of the input data are defined and we check that industry costs equal domestic sales.

Section 2 declares initial values of all variables and assigns values to these variables that are consistent with the base data. By convention, prices are set to 1 — then volumes are equal to initial flow values. Similarly, Section 3 deduces technology and

[30] We are grateful to Erwin Corong for preparing the GAMS implementation.

Table 20.4 Simulation results: 10% improvement in service labor productivity

Macro results	**Variable**		**% change**				
(balance of trade)/GDP (change)	*delB*		0				
Exchange rate (local $/foreign $)	*phi*		0				
Nominal GDP from expenditure side	*wgdpexp*		1.172				
Nominal GDP from income side	*wgdpinc*		1.172				
Real GDP	*xgdp*		1.630				
Aggregates for local final users			***Hou***	***Gov***	***Inv***		
Price indices	*ptot*		0.009	−1.019	−0.281		
Quantity indices	*xtot*		1.162	2.214	1.457		
Nominal expenditure	*wtot*		1.172	1.172	1.172		
Sectoral results			***Agric***	***Mining***	***Manuf***	***Housing***	***Service***
Factor prices	*pfac*	Capital	1.037	1.041	2.652	2.240	5.791
		Labor	−0.427	−0.427	−0.427	−0.427	−0.427
Factor use	*xfac*	Capital	0	0	0	0	0
		Labor	0.732	0.734	1.534	1.330	−2.214
Commodity prices	*p*	*dom*	−0.012	−0.028	0.316	1.350	−2.841
		imp	0	0	0	0	0
Commodity demands	*xdem*	*z dom*	0.369	0.391	0.824	0	4.817
		imp	0.726	1.024	1.732	0	0.588
Exports	*xexp*	*dom*	0.062	0.141	−1.565	−6.484	15.501

Table 20.5 GMS Input file for example GAMS model and simulation

```
$Title        Simple Model
$OFFSYMLIST
$OFFSYMXREF
$EOLCOM !
* Convention: variables (and initial values of variables) are in lowercase,
* other parameters are in upper case
*======================= (1) Set Declaration ========================
* Declare and read Sets and Parameters stored in GDX
Set     ! on data file
 USR      Users
 FAC      Factors
 LOCUSR(USR) Local users
 COM(LOCUSR) Commodities
 IND(COM) Industries
 LOCFIN(LOCUSR) Local final users
 SEC(IND)   Sectors
 SRC      Sources / dom, imp / ;
Parameter    ! on data file
 USE(COM,SRC,USR) Value of inputs
 FACTOR(FAC,IND)  Primary factor costs
 SIGMA(COM)       CES dom-imp elasticity;
$GDXIN input.gdx
$LOADdc USR, FAC, LOCUSR, COM, IND, LOCFIN, SEC, FACTOR, USE, SIGMA
Set   ! Additional Sets
 FIN(USR)    Final users     / Hou, Gov, Inv, Exp /
 EXP(USR)    Exports         / Exp / ;
Alias (IND,i),(COM,c),(SRC,s),(USR,u),(FAC,f),(f,ff),(LOCFIN,lf),(LOCUSR,lu),(FIN,fu);
 FACTOR(f,i)$(FACTOR(f,i)eq 0) = 1E-9; ! alter data to avoid problems
 SIGMA(c) $(SIGMA(c)eq 1) = 1.0001;  !NOTE: CES sigma must not equal 1
Parameter  ! addups of base data
 VALADD0(i)    Factor costs
 COSTS0(i)     Costs
 SALES0(c,s)   Sales
 DIFF(i)       Costs - sales ;
 VALADD0(i)  = SUM[f,FACTOR(f,i)];
 COSTS0(i)   = VALADD0(i) + SUM((c,s), USE(c,s,i));
 SALES0(c,s) = SUM(u, USE(c,s,u));
 DIFF(i)     = SALES0(i,"dom")- COSTS0(i);
DISPLAY VALADD0, COSTS0, SALES0, DIFF, FACTOR;
ABORT$(ABS{SUM[i, DIFF(i)]} GT 0.01)"!!!DATA BALANCE PROBLEM!!!";
*===================== (2) Parameter declarations ====================
Parameter  ! initial values of variables (and variables) are in lower case
 ffac0(f,i)    Factor wage shift
 pfac0(f,i)    Factor prices
 pfac_f0(i)    Factor composite prices
 xfac0(f,i)    Factor use
```

Table 20.5 GMS Input file for example GAMS model and simulation—cont'd

```
 xfac_i0(f)     Total value-weighted factor use
 ffac_i0(f)     Factor wage shift
 xfac_f0(i)     Total factor use by industry
 z0(i)          Industry outputs
 x0(c,s,lu)     Value of inputs
 wtot0(lf)      Nominal expenditure by local final users
 xtot0(lf)      Real expenditure by local final users
 xdem0(c,s)     Total demand for goods
 xcomp0(c,lu)   Quant: final dom-imp composites
 ptot0(lf)      Price indices for local final users
 p0(c,s)        Goods prices
 pcomp0(c,lu)   Price: Intermediate dom-imp composites for local users
 wgdpinc0       Nominal GDP from income side
 wgdpexp0       Nominal GDP from expenditure side
 delB0          Trade balance
 phi0           Exchange rate
 pfimp0(c)      Imported goods prices-foreign $
 xexp0(c)       Export demand
 ftot0(lf)      Absorption parameter
 afac0(f,i)     Factor shifter
 fqexp0(c)      Initial value of export shifter;
*-- Assign initial values
 pcomp0(c,lu) = 1;
 p0(c,s)      = 1;
 pfac_f0(i)   = 1;
 pfac0(f,i)   = 1;
 pfimp0(c)    = 1;
 ptot0(lf)    = 1;
 ffac0(f,i)   = 1;
 ffac_i0(f)   = 1;
 afac0(f,i)   = 1;
 phi0         = 1;
*--- Volumes calculations
 x0(c,s,lu)    = USE(c,s,lu)/p0(c,s);
 xcomp0(c,lu)  = SUM[s,USE(c,s,lu)]/pcomp0(c,lu);
 xexp0(c)      = USE(c,"dom","exp")/p0(c,"dom");
*--- Initial Data for use in calibration
 xfac0(f,i) = FACTOR(f,i)/pfac0(f,i);
 xfac_f0(i) = VALADD0(i)/pfac_f0(i);
 xfac_i0(f) = SUM[i, FACTOR(f,i)/pfac0(f,i)];
 z0(i)      = COSTS0(i)/p0(i,"dom");
 wtot0(lf)  = SUM[(c,s), USE(c,s,lf)];
 xtot0(lf)  = wtot0(lf)/ptot0(lf);
 xdem0(c,s) = SUM[lu, x0(c,s,lu)];
 xdem0(c,"dom") = xdem0(c,"dom") + xexp0(c);
```

(*Continued*)

Table 20.5 GMS Input file for example GAMS model and simulation—cont'd

```
*--- Other initial values
 fqexp0(c)  = xexp0(c)*[p0(c,"dom")/phi0]**(5) ;
 wgdpinc0   = SUM{i, SUM[f,FACTOR(f,i)]};
 delB0      = SUM[c, USE(c,"dom","exp")] - SUM[(c,u), USE(c,"imp",u)];
 wgdpexp0   = SUM[(c,s,lf), USE(c,s,lf)] + delB0;
 delB0      = delB0 / wgdpinc0;
 ftot0(lf)  = wtot0(lf)/wgdpinc0;
*========= (3) Calibration =========
Parameter  ! Calibrate CES dom-imp
 A_ARM(c,s,lu) Armington shares ;
 A_ARM(c,s,lu)$xcomp0(c,lu) = x0(c,s,lu)/xcomp0(c,lu)
                             *[pcomp0(c,lu)/p0(c,s)]**SIGMA(c);
Parameter  ! Calibrate CES primary factor
 SIGMA_FAC(i)    Elasticity in factor use
 ALPHA_F(f,i)    Share of factor in CES value added
 RHO_F(i)        Rho parameter in CES value added
 A_F(i)          Technology parameter in CES value added;
 SIGMA_FAC(i)  = 0.5;
 RHO_F(i)      = (1-SIGMA_FAC(i))/SIGMA_FAC(i);
 ALPHA_F(f,i)  = pfac0(F,I)*xfac0(f,i)**(1+RHO_F(i))/
                  SUM{ff,pfac0(ff,I)*xfac0(ff,i)**[1+RHO_F(i)]};
 A_F(i)        = xfac_f0(i)/SUM[f,ALPHA_F(f,i)*xfac0(f,i)**(-RHO_F(i))]**(-1/RHO_F(i));
Parameter  ! Calibrate Leontief coefficient
 VA_COEF(i)  Value added (Leontief) coefficient for industries
 INT_COEF(c,i) Intermediate (Leontief) coefficient for industries;
 VA_COEF(i)  = z0(i)/xfac_f0(i);
 INT_COEF(c,i) $xcomp0(c,i) = z0(i)/xcomp0(c,i);
Parameter  ! Calibrate Cobb—Douglas for final users
 ALPHA_LF(c,lf) CD Shares of local final users
 A_LF(lf)    CD Shift parameter for local final users;
 ALPHA_LF(c,lf) = [xcomp0(c,lf)*pcomp0(c,lf)]/wtot0(lf);
 A_LF(lf)       = xtot0(lf)/{PROD[c,xcomp0(c,lf)**ALPHA_LF(c,lf)]}
*=========== (4) MODEL ===========
Variable
 pcomp(c,lu)   Price of Composite dom-imp commodity
 p(c,s)        Goods' prices
 pfac_f(i)     Composite factor prices
 pfac(f,i)     Price of Factors
 ptot(lf)      Price indices for local final users
 phi           Exchange rate
 pfimp(i)      Import prices
 xcomp(c,lu)   Composite commodity
 x(c,s,lu)     Intermediate demand
 xfac_f(i)     Total factor use
 xtot(lf)      Total output of local final user
 xfac(f,i)     Factor demand
```

Table 20.5 GMS Input file for example GAMS model and simulation—cont'd

```
 xfac_i(f)    Total factor use
 xtot(lf)     Real expenditure of local final users
 xexp(c)      Export demand
 xdem(c,s)    Total demand for goods
 z(i)         Output of industry
 wtot(lf)     Nominal expenditure by local final users
 wgdpinc      Nominal GDP - income side
 wgdpexp      Nominal GDP expenditure side
 delB         Trade Balance
 afac(f,i)    Factor using technical change
 ffac_i(f)    Factor wage shifter
 ffac(f,i)    Factor wage shifter
 ftot(lf)     Absorption parameter for local final users
 fqexp(c)     Export shifter
 OBJ          NLP objective function;
Equation  ! Dom-imp Block
 E_pcomp      Price of Composite dom-imp commodity
 E_xcomp1     Dom-imp Composite for industries (Leontief)
 E_xcomp2     Dom-imp Composite for local final users (Cobb—Douglas)
 E_x          Intermediate demand ;
 E_pcomp(c,lu)$xcomp0(c,lu).. pcomp(c,lu)*xcomp(c,lu) =e= SUM[s, x(c,s,lu)*p(c,s)];
 E_xcomp1(c,i)$xcomp0(c,i).. xcomp(c,i) =e= z(i)/INT_COEF(c,i);
 E_xcomp2(c,lf).. xcomp(c,lf)*pcomp(c,lf) =e= ALPHA_LF(c,lf)*wtot(lf);
 E_x(c,s,lu).. x(c,s,lu)=e= A_ARM(c,s,lu)*xcomp(c,lu)*[pcomp(c,lu)/p(c,s)]**SIGMA(c);
Equation  ! Industry demands
 E_pA       Industry cost indices
 E_pfac_f   Composite factor prices
 E_xfac     Demand for factors
 E_xfac_f   Composite factor CES;
 E_pA(i,"dom").. p(i,"dom")*z(i) =e= SUM[c,xcomp(c,i)*pcomp(c,i)] + xfac_f(i)*pfac_f(i);
 E_pfac_f(i)..  pfac_f(i)*xfac_f(i) =e= SUM[f, xfac(f,i)*pfac(f,i)];
 E_xfac(f,i)..  xfac(f,i)/xfac_f(i) =e= [afac(f,i)*ALPHA_F(f,i)*pfac_f(i)/pfac(f,i)]
                    **SIGMA_FAC(i)*A_F(i)**(SIGMA_FAC(i)-1);
 E_xfac_f(i)..  xfac_f(i)*VA_COEF(i) =e= z(i);
Equation  ! Final demanders
 E_wtot     Nominal expenditure by local final users
 E_ptot     Price indices for local final users
 E_xtot     Real expenditure by local final users
 E_xexp     Export demand;
 E_wtot(lf)..   wtot(lf) =e= ftot(lf)*wgdpinc;
 E_ptot(lf)..   ptot(lf) =e= [1/A_LF(lf)]*
PROD{c$ALPHA_LF(c,lf),[pcomp(c,lf)/ALPHA_LF(c,lf)]
                    **ALPHA_LF(c,lf)};
 E_xtot(lf)..   xtot(lf)*ptot(lf) =e= wtot(lf);
 E_xexp(c)..   xexp(c) =e= fqexp(c)*[p(c,"dom")/phi]**(-5) ;
Equation  ! Total demand and market clearing
 E_ffac_i     Total factor use
```

(*Continued*)

Table 20.5 GMS Input file for example GAMS model and simulation—cont'd

```
 E_xdemA     Total demand for dom goods
 E_xdemB     Total demand for imp goods
 E_z(i)      Market clearing ;
 E_ffac_i(f)..  xfac_i(f) =e= SUM[i, xfac(f,i)];
 E_xdemA(c)..   xdem(c,"dom") =e= SUM[lu, x(c,"dom",lu)] + xexp(c);
 E_xdemB(c)..   xdem(c,"imp") =e= SUM[lu, x(c,"imp",lu)];
 E_z(i)..       z(i) =e= xdem(i,"dom");
Equation  ! Miscellaneous equations
 E_pB(i)        Import prices
 E_pfac(f,i)    Factor remuneration
 E_wgdpinc      Nominal GDP income
 E_wgdpexp      Nominal GDP expenditure
 E_delB         Trade Balance
 E_OBJ          NLP objective function   ;
 E_pB(i)..      p(i,"imp") =e= pfimp(i)*phi ;
 E_pfac(f,i)..  pfac(f,i) =e= ffac_I(f)*ffac(f,i)* ptot("hou");
 E_wgdpinc..    wgdpinc =e= SUM{i, SUM[f,xfac(f,i)*pfac(f,i)]};
 E_wgdpexp..    wgdpexp =e= SUM[(c,lf)$xcomp0(c,lf), xcomp(c,lf)*pcomp(c,lf)]
                         + SUM[c, xexp(c)*p(c,"dom") - xdem(c,"imp")*p(c,"imp")];
 E_delB..       delB*wgdpinc =e= SUM[c, xexp(c)*p(c,"dom") - xdem(c,"imp")*p(c,"imp")];
 E_OBJ..          OBJ =e= 1;
*--- Initialize levels variables
 xfac_f.l(i)    = xfac_f0(i);
 xcomp.l(c,lu)  = xcomp0(c,lu);
 x.l(c,s,lu)    = x0(c,s,lu);
 xfac.l(f,i)    = xfac0(f,i);
 xtot.l(lf)     = xtot0(lf);
 xexp.l(c)      = xexp0(c);
 xfac_i.l(f)    = xfac_i0(f);
 xdem.l(c,s)    = xdem0(c,s);
 xtot.l(lf)     = xtot0(lf);
 z.l(i)         = z0(i);
 pcomp.l(c,lu)  = pcomp0(c,lu);
 pfac_f.l(i)    = pfac_f0(i);
 pfac.l(f,i)    = pfac0(f,i);
 p.l(c,s)       = p0(c,s);
 p.l(i,s)       = p0(i,s);
 ptot.l(lf)     = ptot0(lf);
 wtot.l(lf)     = wtot0(lf);
 wgdpinc.l      = wgdpinc0;
 wgdpexp.l      = wgdpexp0;
 delB.l         = delB0;
 pfimp.l(i)     = pfimp0(i);
 phi.l          = phi0;
 fqexp.l(c)     = fqexp0(c);
 ftot.l(lf)     = ftot0(lf);
 afac.l(f,i)    = afac0(f,i);
```

Table 20.5 GMS Input file for example GAMS model and simulation—cont'd

```
 ffac_I.l(f)   = ffac_I0(f);
 ffac.l(f,i)   = ffac0(f,i);
 OBJ.l         = 1;
model simple /all/;
*======== (5) Simulation Closure and Shocks ========
* Exogenous Variables in standard closure
 afac.fx(f,i)   = afac0(f,i);
 ffac.fx(f,i)   = ffac0(f,i);
 fqexp.fx(c)    = xexp0(c);
 ftot.fx(lf)    = ftot0(lf);
 pfimp.fx(i)    = pfimp0(i);
 phi.fx         = phi0;
 xfac_i.fx(f)   = xfac_i0(f);
pcomp.fx(c,lu)$(xcomp0(c,lu)=0) = 1;
xcomp.fx(c,lu)$(xcomp0(c,lu)=0) = 0;
option sysout=on;
simple.holdfixed=1;   ! assist check that n.equ = n.endogenous var
simple.iterlim=0;    ! dummy simulation to test calibration
simple.tolinfrep=1E-8;
solve simple maximizing obj using NLP; ! OR: solve simple using mcp;
*--- Closure changes/Swaps
 xfac.fx("capital",i) = xfac0("capital",i);  ! fix industry capital stocks
 ffac.lo("capital",i) = -INF;
 ffac.up("capital",i) = +INF;
 ffac.l("capital",i)  = ffac0("capital",i);
 ffac_I.fx("capital") = ffac_I0("capital");  ! unfix total capital stock
 xfac_i.lo("capital") = -INF;
 xfac_i.up("capital") = +INF;
 xfac_i.l("capital")  = xfac_i0("capital");
*--- SHOCKS
afac.fx("labor","Srv") = 0.98; !Improvement in Service factor productivity
*--- Solve model
simple.reslim=19000;
simple.iterlim=200;
simple.holdfixed=1;
solve simple maximizing obj using NLP; ! OR: solve simple using mcp;
*======== (6) Simulation Results ========
Parameter
 LGDP           Laspeyeres GDP quantity index
 PGDP           Paasche GDP quantity index
 CH_XGDPINC     Percentage change in Fisher (ideal) GDP quantity index
 CH_Z(i)        Percentage change in output of industries
 CH_X(c,s,lu)   Percentage change in intermediate demand
 CH_XFAC(f,i)   Percentage change in factor demand
 CH_PFAC(f,i)   Percentage change in price of factors
 CH_PFAC_F(i)   Percentage change in composite price of factors
 CH_P(c,s)      Percentage change in basic prices
```

(*Continued*)

Table 20.5 GMS Input file for example GAMS model and simulation—cont'd

```
 CH_WGDPINC    Percentage change in nominal GDP income side
 CH_WGDPEXP    Percentage change in nominal GDP expenditure side
 CH_DELB       Ordinary change in ratio nominal trade balance to GDP;
 LGDP = SUM[i,xfac_f.l(i)*pfac_f0(i)] / SUM[i,xfac_f0(i)*pfac_f0(i)];
 PGDP = SUM[i,xfac_f.l(i)*pfac_f.l(i)] / SUM[i,xfac_f0(i)*pfac_f.l(i)];
 CH_XGDPINC = [SQRT(LGDP*PGDP)-1]*100;
 CH_Z(i)                  = (z.l(i)/z0(i)-1)*100;
 CH_X(c,s,lu) $x0(c,s,lu) = (x.l(c,s,lu)/x0(c,s,lu)-1)*100;
 CH_XFAC(f,i) $xfac0(f,i) = (xfac.l(f,i)/xfac0(f,i)-1)*100;
 CH_PFAC(f,i) $pfac0(f,i) = (pfac.l(f,i)/pfac0(f,i)-1)*100;
 CH_PFAC_F(i) $pfac_f0(i) = (pfac_f.l(i)/pfac_f0(i)-1)*100;
 CH_P(c,s)                = (p.l(c,s)/p0(c,s)-1)*100;
 CH_WGDPINC               = (wgdpinc.l/wgdpinc0-1)*100;
 CH_WGDPEXP               = (wgdpexp.l/wgdpexp0-1)*100;
 CH_DELB                  = (delB.l-delB0);
Display CH_Z,CH_XFAC,CH_PFAC,CH_PFAC_F,CH_P,CH_WGDPINC,CH_XGDPINC,CH_WGDPEXP,CH_DELB;
*---Export results to GDX
EXECUTE_UNLOAD "RESULTS," CH_Z,CH_XFAC, CH_PFAC, CH_PFAC_F,CH_P, CH_WGDPINC,
                          CH_XGDPINC, CH_WGDPEXP, CH_DELB;
```

distribution parameters in production functions that are consistent with known prices and quantities.

Section 4 declares the model variables and equations. The variables are set equal to their initial values. A dummy variable, *OBJ*, serves as a maximand in case an optimizing solver is used (even though the system is fully constrained).

Section 5 begins by defining a standard set of exogenous variables (both factors mobile and fixed in aggregate) and performs a trial simulation *without any shock*. The aim is to check that the initial values do indeed constitute a solution. Next, the closure is altered so that industry capital stocks are fixed. A shock is imposed (10% cut in labor needed by services) and the model solved again.

Section 6 uses new and original values to compute a few percentage-change results for reporting. The percentage change in real GDP is computed as a Fisher (ideal) GDP quantity index — giving the same numerical result as the corresponding Divisia index in the GEMPACK model. Results are stored in a .GDX file.

While the GEMPACK modeler automatically receives percentage-change results for *all* variables, the GAMS modeler has to prepare formulae for each percentage change. The path of least resistance is to compute percentage-change results for only a few, summary, variables. In that case, detailed results, which could contain points of interest or even anomalies, may never be viewed.

20.6.4 SIMPLE model in MPSGE

Table 20.6 shows how the SIMPLE model might be represented[31] in MPSGE code. In a GAMS/MPSGE model the code sections for calibration and model equations are replaced by a section written in MPSGE's own syntax. Prior to this, the .GMS file starts with standard GAMS code for declaring sets and data, reading in the data, and defining some consequential totals, such as GDP and output by industry.

The MPSGE syntax extends from the line "`$ontext`" till the line "`$offtext`." An MPSGE model revolves around three types of entity:

- Constant-returns-to-scale producers, listed under `$sectors`.
- Goods and primary factors (defined by a unique price), defined in the `$commodities` section.
- Consumers (either government or private), in the `$consumers` section

The `$Prod` section compactly expresses the production technology — it is the textbook equivalent of Figure 20.1. It says that each industry is the sole producer of a domestic commodity, price $P(j)$. Domestic and imported commodities are combined to form a CES composite (the "`tl`" nest), while primary factors are likewise combined in the nest "`va`." At the top level, production is Leontief ("`s:0`") in composite commodities and composite value-added.

Similarly, the second `$demand` block says that each of the "`fd`" set (`hou`, `gov`, `inv`) is a Cobb-Douglas consumer (`s:1`) of composite commodities, which are again CES combinations of domestic and imported commodities.

These statements would suffice for the simplest type of MPSGE model. Unfortunately the SIMPLE model defined in previous sections breaks the MPSGE paradigm in three respects:

- Export demand curves are not flat (the MPSGE expectation), but are of a constant elasticity.
- Not all the primary factors are perfectly mobile (the MPSGE expectation); capital may be sector-specific.
- The value of absorption is not equal to factor earnings (the MPSGE expectation), but is rather constrained by a rule that the trade balance must move in proportion to GDP.

Luckily it is always possible (with some ingenuity and a good understanding of MPSGE) to accommodate such unorthodoxies. The `$auxiliary` and `$constraint` sections do this job. For example, the first `$constraint` defines the constant elasticity export demand curves.

Two types of capital are defined: mobile capital has a common price [W("capital")], while rents vary for sector-specific capital [endowment $ssk(j)$ with price $RK(j)$].

[31] The MPSGE implementation is due to Tom Rutherford.

Table 20.6 GMS Input file for example MPSGE model and simulation

```
$title A Simple General Equilibrium Model
* Next line enables end-of-line comments following #
$eolcom #
set
       i     Goods and sectors  # COM, or IND
       fd    Components final demand /hou,gov,inv/,  # LOCFIN
       f     Primary factors /labor,capital/, # This capital is mobile  # FAC
       src   Sources /dom,imp/;  # SRC
$gdxin input.gdx
$load i
parameter
       use(i,src,*)  Value of inputs,
       factor(f,i)   Factor demands,
       sigma(i)      CES domestic-import elasticity;
$loaddc use factor sigma
alias (i,j);
parameter
       output(i)     Aggregate output     COSTS(c)
       expend(fd)    Final demand expenditure  COSTS(L)
       export(i)     Benchmark exports      USE(c:dom:exp)
       epsilonx(i)   Export demand elasticity    5.0
       bopdef        Base year current account deficit
       sigmaf(i)     Elasticity of substitution among factors  one half
       prd(f,j)      Productivity index  AFAC(f:i)
       ssk(j)        Sector-specific capital
       endow(f)      Primary factor endowment vector  SUM(i:FACTOR(f:i))
       gdp0          Benchmark gross domestic product  VGDP  ;
prd(f,j)=1;
output(j)=sum((i,src),use(i,src,j))+sum(f,factor(f,j));
export(i)=use(i,"dom","exp");
expend(fd)=sum((i,src),use(i,src,fd));
epsilonx(i)=5;
sigmaf(i)=0.5;
ssk(j)=0; # start with no sector-specific capital
bopdef=sum((i,j),use(i,"imp",j))+sum((i,fd),use(i,"imp",fd))-sum(i,export(i));
endow(f)=(sum(j,factor(f,j)));
gdp0=sum(f,endow(f));
$ontext
$model:simple
* Comment within MPSGE code
$sectors:
   Y(j)      ! 1-based index of production  ZINDEX
$commodities:
   P(j)      ! Price of domestic output  p(c:dom)
   PFX       ! Exchange rate (Dollar domestic per foreign dollar) PHI
* There are 2 kinds of capital -
```

Table 20.6 GMS Input file for example MPSGE model and simulation—cont'd

```
* Mobile has price W("capital") and fixed capital in sector j has price RK(j)
* In SIMPLE.TAB factor price is pfac(f:i) which might depend on using industry i.
        W(f)$endow(f)  ! Factor price   PFAC(f:i)
        PGNP           ! Price index for factor income
        RK(j)$ssk(j)   ! Rental rate on sector-specific capital  PFAC(capital:i)
$consumers:
        GNP            ! Aggregate national income
        RA(fd)         ! Expenditure by representative agent  WTOT(L)
$auxiliary:
        X(i)           ! Export quantity   XEXP(c)
        VX             ! Value of exports (Foreign currency)  COSTS(exp) in domestic dollars
        GDP            ! 1-based index of the value of GDP in foreign dollars
$Prod:Y(j)   s:0 va:sigmaf(j) i.tl:sigma(i)
        o:P(j)        q:output(j)
        i:W(f)        q:(factor(f,j)/prd(f,j)) p:prd(f,j)  va:
        i:RK(j)       q:ssk(j)       va:
        i:PFX#(i)     q:use(i,"imp",j)    i.tl:
        i:P(i)        q:use(i,"dom",j)    i.tl:
$report:
        v:wd(f,j)     i:W(f) prod:Y(j)
        v:VINTIMP(j)  i:PFX#(i) prod:Y(j)
$demand:GNP
* Next two lines add up to primary factor income
        e:W(f)        q:endow(f)
        e:RK(j)       q:ssk(j)
* Next 2 lines seem to cancel each other out
        e:P(i)        q:(-1)         r:X(i)
        e:PFX         q:1            r:VX
* Next says there is a foreign gift of PFX*bopdef*GDP local dollars
        e:PFX         q:bopdef       r:GDP
        d:PGNP
$demand:RA(fd)   s:1 i.tl:sigma(i)
* Next says income(fd)=PGNP*expend(fd) for each of the 3 fd's
        e:PGNP        q:expend(fd)
* Next 2 lines say all fd have armington demands for import v domestic with elasticity sigma
* and that elasticity is 1 between different commodities
        d:P(i)        q:use(i,"dom",fd)    i.tl:
        d:PFX#(i)     q:use(i,"imp",fd)    i.tl:
$report:
        v:VFINIMP(fd) d:PFX#(i) demand:RA(fd)
$constraint:X(i)
        X(i) =e= export(i)*(PFX/P(i))**epsilonx(i);
$constraint:VX
        PFX*VX =e= sum(i, P(i)*X(i));
$constraint:GDP
* GDP is a 1-based index of the value of GDP in foreign dollars
* Actual GDP is LHS of next equation.
```

(Continued)

Table 20.6 GMS Input file for example MPSGE model and simulation—cont'd

```
        GDP*gdp0*PFX =e= sum(f, W(f)*endow(f)) + sum(j, RK(j)*ssk(j));
$offtext
$sysinclude mpsgeset simple
* NOTE. Foreign gift divided by actual GDP is equal to
*  PFX*bopdef*GDP/GDP*gpd0*PFX = bopdef/gdp0 which is constant
GDP.L = 1;
X.L(i) = export(i);
X.FX(i)$(export(i)=0) = 0;
VX.L = sum(i, X.L(i));
*       Assign a numeraire to simplify comparison with the GEMPACK solution:
PFX.FX = 1;
simple.iterlim = 0;
simple.workspace = 50;
$include simple.gen
solve simple using mcp;
abort$(simple.objval>1e-2) "Benchmark without SSK inconsistent.";
*       Swap the closure and verify
ssk(j) = factor("capital",j);
factor("capital",j) = 0;
endow("capital") = 0;
simple.iterlim = 0;
simple.workspace = 50;
$include simple.gen
solve simple using mcp;
abort$(simple.objval>1e-2) "Benchmark with SSK inconsistent.";
display VINTIMP.L;
parameter summary  Summary of model results ;
summary("ActualGDP", "initial") = sum(f, W.L(f)*endow(f)) + sum(j, RK.L(j)*ssk(j)) ;
summary("VIMP","initial") = sum(j, VINTIMP.L(j)) + sum(fd, VFINIMP.L(fd)) ;
summary("VEXP","initial") = sum(i, P.L(i)*X.L(i)) ;
summary("BOP", "initial") = summary("VEXP","initial") - summary("VIMP", "initial") ;
summary("BOPRatio","initial") = summary("BOP","initial")/summary("ActualGDP","initial") ;
*       Simulate a productivity shock for labor inputs to the service sector:
prd("labor","srv") = 1/0.98;
*       First simulation is based on mobile capital:
factor("capital",j) = ssk(j);
endow("capital") = sum(j,ssk(j));
ssk(j) = 0;
simple.iterlim = 20000;
$include simple.gen
solve simple using mcp;
parameter    impact Economic impact;
impact("Y%",j,"mobileK") = 100*(Y.L(j)-1);
impact("X%",j,"mobileK")$export(j) = 100*(X.L(j)/export(j)-1);
impact("P%",j,"mobileK") = 100*(P.L(j)/PFX.L-1);
impact("P%",f,"mobileK")$endow(f) = 100*(W.L(f)/PFX.L-1);
impact("RK%",j,"mobileK")$ssk(j) = 100*(RK.L(j)/PFX.L-1);
```

Table 20.6 GMS Input file for example MPSGE model and simulation—cont'd

```
summary("ActualGDP","mobileK") = sum(f, W.L(f)*endow(f)) + sum(j, RK.L(j)*ssk(j));
summary("VIMP","mobileK") = sum(j, VINTIMP.L(j)) + sum(fd, VFINIMP.L(fd));
summary("VEXP","mobileK") = sum(i, P.L(i)*X.L(i));
summary("BOP","mobileK") = summary("VEXP","mobileK") - summary("VIMP","mobileK");
summary("BOPRatio","mobileK") = summary("BOP","mobileK")/summary("ActualGDP","mobileK");
*       Second simulation is based on sector-specific capital:
ssk(j) = factor("capital",j);
factor("capital",j) = 0;
endow("capital") = 0;
simple.iterlim = 20000;
$include simple.gen
solve simple using mcp;
impact("Y%",j,"SSK") = 100*(Y.L(j)-1);
impact("X%",j,"SSK")$export(j) = 100*(X.L(j)/export(j)-1);
impact("P%",j,"SSK") = 100*(P.L(j)/PFX.L-1);
impact("P%",f,"SSK")$endow(f) = 100*(W.L(f)/PFX.L-1);
impact("RK%",j,"SSK")$ssk(j) = 100*(RK.L(j)/PFX.L-1);
summary("ActualGDP","SSK") = sum(f, W.L(f)*endow(f)) + sum(j, RK.L(j)*ssk(j));
summary("VIMP","SSK") = sum(j, VINTIMP.L(j)) + sum(fd, VFINIMP.L(fd));
summary("VEXP","SSK") = sum(i, P.L(i)*X.L(i));
summary("BOP","SSK") = summary("VEXP","SSK") - summary("VIMP","SSK");
summary("BOPRatio","SSK") = summary("BOP","SSK")/summary("ActualGDP","SSK");
option impact:3:1:1;
display impact;
option summary:5:1:1;
display summary;
EXECUTE_UNLOAD "RESULTS," Impact;
```

Changing the closure from mobile to fixed capital is accomplished by altering the database: moving all capital earnings to accrue to the fixed type of capital.

The implementation of the balance of trade constraint is more complicated. The first `$demand` block states that absorption is, as usual, equal to income; however, that income includes not only primary factor income, but also a gift from abroad of value PFX ∗ *bopdef* ∗ GDP, where *bopdef* is the initial BOT/GDP[32] ratio.

Even with these complications the MPSGE code is only half the length of the corresponding straight GAMS code for calibration and model equations. Indeed, it does *more* than the conventional GAMS code that it replaces:

- The implicit calibration is done in the most efficient possible manner.
- Equations and variables for intermediate use are effectively substituted out, reducing system size.
- While both GEMPACK and straight GAMS implementations assume (not unusually) that all outputs and prices remain strictly positive, MPSGE sets up a full system of

[32] Note the later `$constraint` stating that GDP is a 1-based index!

complementarities which allows for inactive sectors and slack markets. This considerably extends the scope of the system.[33]

The remainder of the GMS file is ordinary GAMS code. There are three simulations: a test simulation to reproduce the benchmark data and two simulations of improved Services labor productivity — the first with mobile capital, the second with sector-specific capital.

A small selection of percentage-change results from the last two simulations are stored in the matrix "`impact`." The formulae presuppose a knowledge of MPSGE's default rules for price and quantity normalization. For example, sector outputs are initially 1, so $100 * (Y \cdot L(j) - 1)$ is the percentage change in outputs. Similarly most prices are initially 1.

Similarly some levels results are stored in the matrix "`summary`." In this case, the role of "`summary`" is mostly to check that the BOT/GDP ratio does in fact remain constant.

20.6.6 All three give the same results

As expected, the same results are generated by GEMPACK, GAMS and MPSGE. You can check this by dowloading the models from http://www.monash.edu.au/policy/archivep.htm#TPMH0103.

20.6.7 Understanding the results

Understanding the results means working out how the shocks, together with the starting data and the theory of the model, produce changes in endogenous variables. For each important endogenous result, you want to understand both the sign and the approximate magnitude (e.g. why is it 3 and not 0.1).

This is a complicated topic that we cannot hope to cover in detail in this chapter. To give some of the flavor, consider the following question drawn from our example simulation.

Question Does improved service-worker productivity mean service workers get fired? How come there is a 10% productivity increase, but only a 2% job loss in the Service industry?

This question is motivated by the result that employment in the Service industry falls by 2.214% in the simulation. This is variable *xfac*("Labor," "srv") in the GEMPACK and GAMS implementations.

It turns out that the linear or percentage-change form of the equations is usually more helpful for understanding results than the levels form. GEMPACK provides a viewing program, AnalyseGE, which streamlines this kind of analysis. Below we illustrate this to (partly) answer the question above.

[33] Both GEMPACK and straight GAMS allow complementarities to be set up — at the cost of more complex code.

The equation in SIMPLE.TAB that "explains" the variable *xfac* is the CES factor demand equation:

```
E_xfac # Factor demands # (all,f,FAC)(all,i,IND)
 xfac(f,i) = z(i) + afac(f,i) - 0.5*[pfac(f,i) + afac(f,i) - pfac_f(i)];
```

You can ask AnalyseGE to evaluate the variable *xfac* (it tells you −2.214 for the "Labor," "srv" part) and to decompose the right-hand side. The decomposition (when $f =$ "Labor" and $i =$ "srv") shows you the table:

```
z  4.817
afac  -10.000
e1_pfac  3.359
Total  -1.824
```

The shock value *afac* is directly in this equation. Productivity has increased by 10%. Now overall activity "*z*" in this industry has increased by 4.817%. The *e1_pfac* value of 3.359 is the value of the last, relative-price, term $-0.5 * [\ldots]$ on the right-hand side of the equation. The [...] term is about −6.7 because the effective cost of labor in this industry, *pfac*(*f*,*i*) + *afac*(*f*,*i*), falls faster the average cost of factor inputs *pfac_f*(*i*) in this industry, resulting in a boost of 3.359% to industry employment. That boost arises because with capital fixed (and ignoring technical change), an increase in output must raise the labor/capital ratio and thus output-per-worker falls.

Although the productivity increase would seem to decrease employment by 10%, this is offset by the increased employment needed because activity in the industry increases by 4.8% and because productivity falls with output, resulting in a further 3.36% increase in employment. Overall employment then falls only by about 2%. Notice that the elasticity value of 0.5 enters into these calculations. We may imagine that if the elasticity of substitution between capital and labor was 1 (rather than 0.5) the net effects on service employment might be positive.[34]

The "Total" result shown by AnalyseGE, −1.824 (which is obtained by adding the *z*, *afac* and *e1_pfac* components above) is not exactly equal to the result −2.214 for *xfac*("Labor," "srv"), since the linearized equations in the TAB file are not satisfied exactly by the accurate results.[35]

The above analysis could also be carried out by GAMS users. However, probably they would need to explicitly add into the post-solution analysis of their .GMS file the linearized version of equation *E_xfac* and formulae to calculate the different parts of the decomposition discussed above.

[34] However, capital rents and Services output will change when we change the elasticity. Rerunning the simulation with elasticity value 1, we found that service employment was −0.060 (still slightly negative).

[35] See, e.g. http://www.monash.edu.au/policy/gp-lineq.htm.

Table 20.7 Solution time (seconds) versus number of sectors

	Sectors								Elasticity[a] of time with respect to sectors
	100	150	200	250	300	350	400	500	
GEMPACK exe	1	2.2	3.7	5.8	8.6	11.7	16.4	26.0	2.1
GEMSIM	1.88	4.0	7.1	11.2	16.2	22.4	29.7	49.3	2.3
GAMS MCP	2.4	8.1	31.5	111.5	324.6	979.6	1763.0	7550.4	6.5
GAMS NLP	1.6	3.9	8.6	14.0	115.5	235.2	400.7	1107.0	4.6
MPSGE	2.1	7.1	18.2	40.1	78.9	138.5	225.9	524.3	3.8

[a]This is equal to LOG(T500/T400)/LOG(500/400, where T500 and T400 are the times for the 500- and 400-sector runs. Note that this number is the slope between 400 and 500 sectors in the log-log graph in Figure 20.3.

20.7 CURSE OF DIMENSIONALITY

Policy-oriented CGE modelers are constantly tempted to increase the numbers of commodities (C), industries (I), regions (R) or household types (H) distinguished in their models. More sectors allow for greater technological detail, distinguishing, for example, between several technologies for producing electricity. Again, to model a tariff reform, where benefits arise from reducing the dispersion of tax rates, it helps to distinguish commodities finely; ideally at the HS (Harmonized System) six-digit level (HS6) level by which tariff rates vary. Dividing households by income decile, urban/rural and ethnicity, we can easily arrive at $H = 100$. It may be politically unacceptable to reduce the number of regions, R, by combining, say, the Dakotas, even if simulation results were not much affected.

However, this detail — the special strength of CGE — comes at a high price: it can prohibitively increase the computing resources (memory and especially time) needed to solve the model. The size of equations and variables is often the *product* of various dimensions: household demands may be dimensioned $C * H * R$, while intermediate demands might be size $C * I * R$, i.e. would increase as the square of the number of sectors.

Further, theory suggests that the cost of solving a linear system (an important subtask in solution algorithms) follows the *cube* or the *square* of the number of equations. Hence, we may guesstimate that doubling the number of commodities and industries in a model may increase solution time by a factor of 2^5 (between 2^4 and 2^6).

To illustrate this idea we solved the SIMPLE model presented earlier with databases of several sizes, and using several solution methods. The resulting solution times are shown in Table 20.7.[36] The first two rows use GEMPACK in conjunction with the SIMPLE.TAB of

[36] The models were solved on one core of an Intel Q9550 CPU, under 64-bit Vista. The databases were sparse — about 70% zeroes. All material needed to repeat the simulations is downloadable: item TPMH0103 at http://www.monash.edu.au/policy/archivep.htm.

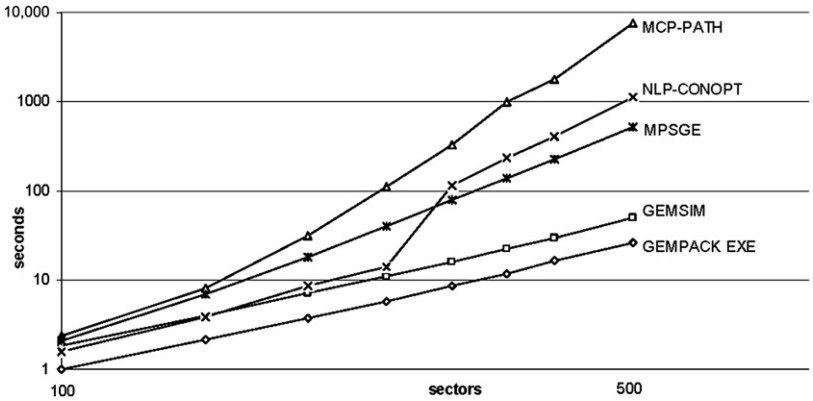

Figure 20.3 Solution time versus number of sectors: Log-log scale.

Table 20.2. For the first row, the GEMPACK program TABLO has written a model-specific Fortran program, SIMPLE.FOR, which has been compiled using an optimizing compiler to produce SIMPLE.EXE. This is about twice as fast as the alternative GEMPACK approach (row 2), which uses an interpreter, GEMSIM. The next two rows use the "straight GAMS" code of Table 20.5. Row 3 uses the PATH MCP solver while row 4 uses the CONOPT NLP solver. Finally, row 5 shows times[37] for the MPSGE specification of Table 20.6.

While during the 1980s the solution time for a 200-sector model might well allow the modeler to enjoy a night's sleep, today's tools do not allow enough time to make a cup of coffee! Only as the number of sectors approaches 400 do solution times become a real concern — and which sane CGE modeler could want more than 200 sectors?

In fact, multiregional and multitemporal CGE models, which comprise a collection of interlinked models rather like SIMPLE for each region and time period, can easily reach dimensions much larger than the 500-sector SIMPLE. Such models solve slowly. Modelers are forced to adopt compromises to keep compute times within limits.

Very likely a skilled practitioner could "tweak" any of these implementations to halve solution times. Nonetheless, Table 20.7 well illustrates how moving from, say, 400 to 500 sectors dramatically lengthens solution time. The point is even clearer in Figure 20.3, which graphs the same figures on a log-log scale. Each solution method yields a straight-line graph,[38] the slope of which shows the elasticity of solution time with respect to number of sectors. That elasticity, measured over the 400—500 interval, is shown in the final column of Table 20.7 and is markedly lower for the two GEMPACK implementations.

37 MPSGE times have been halved, because the .GMS file specified two simulations.
38 For CONOPT, there is a break in the line, between 250 and 300 sectors.

One reason that GEMPACK scales well in this example is that high-dimension equations of size COM * IND have been automatically substituted out.[39] Hence, the final (after-substitution) number of equations follow the number of sectors (rather than its square).[40] MPSGE achieves a similar effect.

A "straight GAMS" user could enjoy a similar speedup by manually substituting out high-dimension equations. For example (see Section 20.4.2.1), the equation:

```
x(c,s,u)=A_ARM(c,s,u)*xcomp(c,u)*[pcomp(c,u)/p(c,s)]**SIGMA(c);
```

could be used to substitute out the left-hand side variable *x*. That strategy could lead to complicated, hard-to-maintain, model code. Alternatively the GAMS modeler may avoid complicated mechanisms that are both commodity- and sector-specific. For example, import-domestic substitution might be modeled as commodity-specific but not user-specific — greatly simplifying the model, without, usually,[41] much affecting results.

20.8 CHECKING AND DEBUGGING MODELS

How can we be sure that a CGE model is correctly coded or that it is using reasonable data? Careful "eyeballing" of the code is too time-consuming to do often. Can the software help us?

Standard techniques for checking[42] models include:

- Balance and sign checks on input data.
- In GAMS only, a check that the calibrated or benchmark prices and quantities satisfy model equations (in GEMPACK there are no levels prices and quantities).
- Standard experiments, such as a numéraire shock, which should increase all prices and values without affecting quantities.
- Balance checks on updated data or a check that a redundant accounting constraint is satisfied by the results (often termed the "Walras" check).

Both GAMS and GEMPACK have the ability to apply such checks and to abort simulations with an appropriate message if conditions are not satisfied. Problems are often identified in this way. Nonetheless, such checks do not screen out all bugs:

- Under pressure, modelers might comment out the checking lines and forget to reinstate them later.

[39] Substitution is not the whole story. When we used GEMPACK to solve the uncondensed model, the elasticities increased (as expected), but both were still no higher than 2.5. (The solve times were about three times longer than for the condensed model.)

[40] For $N = 500$, the original number of equations was just over 1 million; substitution eliminated 75% of these.

[41] The exception would be when some users of an imported good have a markedly different import share from other users. For example, in analyzing textile tariff reductions we may observe that the domestic Textile, clothing and footwear (TCF) industry is a particularly intensive user of imported TCF. As production is globalized we see these cases more often. To model them properly, import-domestic substitution must be user-specific.

[42] Dixon and Rimmer in Chapter 19 of this Handbook further discuss the checking of models.

- Many faults might evade automatic checks, including wrongly signed elasticities, or inconsistent ordering of sectors or regions in different data sources.

To detect such problems and locate their source, there is really no substitute for detailed examination of data and results. GEMPACK attempts to make this process as easy and painless as possible. There is a specialized viewer for data (ViewHAR), another for results (ViewSOL) and a third (AnalyseGE) that allows viewing of data or results (or any function of the two) by clicking on coefficients or variables in the model code. These viewers are quite sophisticated; ViewSOL, for instance, is able to load one or two time series of model solutions and display these as year-on-year or cumulative changes or to display the difference between control and perturbed scenarios.

20.9 COMPARING FEATURES OF GAMS AND GEMPACK

GEMPACK was designed to solve CGE models; it performs a more specialized task than GAMS. The .TAB file specifying GEMPACK model describes, in differential form, the model's non-linear equations: the software uses those equations to do numerical integration and find one new levels solution. It is not designed as a general-purpose programming tool.

GAMS, as well as interfacing to various optimizers and equation solvers, provides a more feature-rich computer language. It allows loops and other control constructs, include files, macro preprocessing, and, crucially, the ability to execute other programs from within a GAMS script. This makes GAMS immensely versatile.

For example, it would be fairly easy, within GAMS, to repeat 30 times the same simulation with different values of the capital-labor substitution elasticity and to have another program such as Excel graph some results. The task could be completely automated from within one .GMS file.

To do the same job with GEMPACK, you would use another scripting language (probably the Windows BATch language) to run 30 simulations, to call other GEMPACK programs to collate and reformat the results, and finally to invoke Excel. This means that a serious GEMPACK user needs to master another skill (batch scripting).

GAMS also allows for a user to add features not provided by, or envisaged, by the developers. For example, suppose you want to define and use repeatedly a function not explicitly provided. A programmer can do this with GAMS,[43] but GEMPACK users must ask the developers to add a new function.

[43] http://www.gams.com/docs/extfunc.htm.

By giving access to low-level features of the software, GAMS makes it possible and easy for power users to do things not explicitly provided for by the developers. They can also write so-called libinclude files which give high level functionality to other users.[44]

By contrast GEMPACK makes some complex tasks easy even for computer novices. Examples are:

- *Recursive-dynamic models.* While it is possible to implement such models in GAMS, GEMPACK provides the RunDynam windows interface, which does all the book-keeping to organize the runs in sequence, and to assemble and report the results. GAMS users must keep track of the sequence of runs needed to cover the years in sequence and to organize the results. Provided your model fits within the (wide) range of cases allowed, using RunDynam is the easier route. However, if you want to do something outside that framework, you may find that easier with GAMS.
- *Systematic sensitivity analysis.* The model must be run many different times, each time with different parameters and/or shocks and the results assembled to report means and standard deviations across the runs. A power GAMS user can write a procedure to vary the parameters or shocks in the appropriate way and keep track of all the relevant files and runs to do this. GEMPACK provides a windows interface (in RunGTAP and RunGEM) for doing all of this. The user just needs to point and click the few details and the software does the rest, but of course there are restrictions on the sorts of variations allowed. If your task falls outside those allowed, GEMPACK users would also need to handle all the details themselves.
- *Decomposition of results due to shocks.* This tells us, for example, how much of the US GDP change due to a bilateral tariff reform arises from the US tariff reduction and how much from the EU tariff reduction. The theory is fairly simple (see Harrison *et al.*, 2000) and may be implemented by GEMPACK users with two or three lines of code at modest additional computational cost. Power GAMS users can carry out this decomposition in GAMS[45], with a bit more effort.

Other differences between GAMS and GEMPACK include the following.

20.9.1.1 Optimization

GAMS is first and foremost an optimization package, with CGE modeling arising as an unexpected use. Thus, it provides state-of-the-art capability for optimization.

CGE modelers may want to solve optimization problems in simulations which involve choice of optimum shock (e.g. optimal tax). However, optimization problems

[44] See, e.g. the GAMS2HAR routines that allow conversion of GAMS text data to GEMPACK Header Array form, at http://www.monash.edu.au/policy/gp-g2har.htm.

[45] See, e.g. http://www.mpsge.org/mainpage/hhp_gtm.htm or Böhringer and Rutherford (2000).

also arise in procedures for preparing or massaging data to use in CGE models. For example, minimum entropy methods might be used to balance a SAM.

GEMPACK offers no direct features for optimization. However, by using the first-order conditions directly, it is possible to solve simpler optimization problems.

20.9.1.2 Complementarities

GAMS comes with powerful algorithms (such as PATH) for explicitly modeling complementarities. GEMPACK also has explicit capabilities for handling these, which work well for most problems arising naturally in policy models, but the algorithms used by GEMPACK make this a less natural vehicle for such problems.

20.9.1.3 Distributing models to non-licensees

GEMPACK allows for models to be distributed as stand-alone executable images (e.g. GTAP.EXE) which may be run without any license. Thus, RunGTAP, an IDE for running the GTAP model, is available without cost. Only GAMS licensees can solve GAMS models.

20.9.1.4 Training and documentation

Several complementary techniques are used to teach CGE modeling, either for GAMS or GEMPACK:

- *The apprenticeship system*. Most CGE modelers would learn their craft during graduate study or at work.
- *Training courses.* Intensive (typically 1-week) courses are offered, often for beginners though also for more advanced users. Many of the GAMS courses are offered by third parties, rather than directly by the GAMS corporation.
- *Supplied documentation*. Both GAMS and GEMPACK have produced printed documentation (see Brooke *et al.*, 1992; and see *GEMPACK Documents 1—9* at http://www.monash.edu.au/policy/gpdoc.htm), but it is difficult to make such documentation comprehensive, up-to-date and accessible. The printed manuals are supplemented by (indeed, mostly replaced by) online help and by much instructional material from the respective websites.
- *On-line forums*, such as the GAMS and GEMPACK mailing lists. The GAMS list is particularly active.

20.10 CONCLUDING REMARKS

20.10.1 What are the computational challenges?

Our survey has been somewhat retrospective, showing how standard software packages have enabled the ideas of CGE pioneers of the 1970s and 1980s to be implemented in today's more standardized software environment. That standardization has facilitated the spread of standard CGE models within and beyond academic circles. However, what are today's newer ideas and techniques, and will CGE software evolve to make them practical?

- The combination of dynamic programming with explicit representation of uncertainty is computationally challenging. The two ideas are united in the area of dynamic stochastic general equilibrium (DSGE). Currently, DSGE models contain too few sectors to qualify as CGE models, but this may change. The DYNARE software (see http://www.dynare.org) is being used by some modelers, especially those doing modeling for central banks (http://www.dsge.net). It is tailored to DSGE and overlapping generations (OLG) models, that include forward-looking behavior. Tools for carrying out global sensitivity analysis are being incorporated into DYNARE.[46]
- Agent-based modeling is an increasingly popular paradigm, which may be amenable to a high level of parallelization in computing.
- Some theorists complain that multiple equilibria are endemic in general equilibrium models, yet CGE practitioners rarely encounter them — partly because of the usual form of CGE models. However, approaches such as game theory can easily lead to multiple local equilibria, which standard CGE software does not easily identify.

Practitioners who experiment with these ideas use a variety of software, including MATLAB and even Excel. Some DSGE modelers still write Fortran code.

The Institute of Computational Economics at Argonne and Chicago carries out an active program encouraging interaction between economists and algorithm specialists. The Institute provides regular training opportunities for professionals in both areas (see http://ice.uchicago.edu/info.htm for more details).

Another challenge is to link together several models, e.g. to link a world model such as GTAP with a more detailed model of one country or, instead, to link a CGE model with a climate model and perhaps also with an agronomist's model relating farm output to climate. The collection of models is supposed to jointly determine a range of economic and other results. Practical problems include different software and data formats for the different models, model databases that may not be mutually consistent, and the likelihood that some variables are modeled independently and differently in two or more models. The typical solution is to construct a master program that will manage communication between the models and iteratively solve them in turn until all results are consistent. The master program is likely to be a "home-brewed" one-off that only ever runs on the computer network where it was developed — rather like the first CGE models. There is scope, then, to develop both a general conceptual framework for model linking and software that is able to coordinate simulations from a range of model types.

[46] Both GAMS and GEMPACK include capabilities for solving forward-looking models, but have nothing like the global sensitivity tools in DYNARE.

20.10.2 Transferring models between the software platforms

As described in Section 20.1, early CGE modelers worked alone or in small groups using customized solution routines and model-specific code. It was therefore difficult to replicate model results and model documentation had to be based on trust.

Great steps have been made since those early days. Standardized modeling packages have lowered the entry costs to CGE and enormously increased the transparency of models. The establishment of CGE modeling teams in government departments has created a job market for CGE modelers, who will have transferable skills.

Nonetheless, there is still some distance to go.

- In Section 20.1, we said: "Most models implemented with one of these could, at least in principle, be implemented with either of the other two and yield the same results." However, when we tried this with the SIMPLE model in Section 20.6, we were surprised how difficult it was in practice.
- It may take 5000 lines of code to implement a large CGE model. To translate such a model from GAMS to GEMPACK or *vice versa* would be a large task. To understand such a model in detail rather than in outline would similarly require some months of work, even for a very experienced modeler who normally works with the relevant software (GAMS or GEMPACK).
- Most experienced CGE modelers are comfortable and fluent with only one modeling system, in which they have invested considerable time and effort. They are naturally reluctant to contemplate changing. Modelers become attached to their accustomed software, notation and equations and closure.
- While it is easy to translate data between the .HAR and .GDX file formats, there is no automated way to translate model code from GAMS to GEMPACK or *vice versa*. A possible solution would to be represent CGE models in a third, more abstract language from which either GAMS or GEMPACK code could be automatically generated. The MPSGE language approaches this ideal. However, such systems impose a template or pattern on the model. Trend-leading CGE modelers, who by definition experiment with model specifications outside the standard recipe, may find the template annoying or constricting.

Thus, as long as CGE modeling continues to develop, it is likely that a diversity of software approaches will be used.

REFERENCES

Adelman, I., Robinson, S., 1978. Income Distribution Policy in Developing Countries: A Case Study of Korea. Stanford University Press, Stanford, CA.

Bisschop, J., Meeraus, A., 1982. On the development of a general algebraic modeling system in a strategic planning environment. Math. Progr. Study 20, 1–19.

Böhringer, C., Rutherford, T., 2000. Decomposing the Cost of Kyoto: A Global CGE Analysis of Multilateral Policy Impacts. Discussion Paper 00–11. ZEW (Center for European Economic Research), Mannheim.

Brooke, A., Meeraus, A., Kendrick, D., 1992. GAMS: A User's Guide. Release 2.25. The Scientific Press, San Francisco, CA.

Codsi, G., Pearson, K.R., 1986. GEMPACK Documents 2, 3, 5–8 and 11–18. IMPACT Project, Melbourne.

Codsi, G., Pearson, K.R., 1988. GEMPACK: General-purpose software for applied general equilibrium and other economic modellers. Comput. Sci. Econ. Manag. 1, 189–207.

Condon, T., Dahl, H., Devarajan, S., 1987. Implementing a Computable General Equilibrium Model on GAMS: The Cameroon Model. Report DRD290. World Bank, Washington, DC.

Dixon, P.B., 1975. The Theory of Joint Maximization. Contributions to Economic Analysis 91. North-Holland, Amsterdam.

Dixon, P.B., Parmenter, B.R., Ryland, G.J., Sutton, J., 1977. vol. 2 of the First Progress Report of the IMPACT Project. ORANI, A General Equilibrium Model of the Australian Economy: Current Specification and Illustrations of Use for Policy Analysis. Australian Government Publishing Service, Canberra.

Dixon, P.B., Parmenter, B.R., Sutton, J., Vincent, D.P., 1982. ORANI: A Multisectoral Model of the Australian Economy. Contributions to Economic Analysis 142. North-Holland, Amsterdam.

Drud, A., 1989. Hercules—a modeling system with knowledge about economics. Comput. Econ. 2, 83–99.

Drud, A., Kendrick, A., Meeraus, A., 1986. HERCULES: A System for Development of Multisectoral Economy-wide Models. World Bank Discussion Paper DRD 169. World Bank, Washington, DC.

Duff, I.S., Reid;, J.K., Erisman, A.M., 1989. Direct Methods for Sparse Matrices. Oxford University Press, Oxford.

Harrison, W.J., Horridge, J.M., Pearson, K.R., 2000. Decomposing simulation results with respect to exogenous shocks. Comput. Econ. 15, 227–249. Available at: http://www.monash.edu.au/policy/elecpapr/ip-73.htm.

Hertel, T. (Ed.), 1997. Global Trade Analysis: Modeling and Applications. Cambridge University Press, Cambridge.

Hertel, T.W., Horridge, J.M., Pearson, K.R., 1992. Mending the family tree: A reconciliation of the linearized and levels schools of AGE modelling. Econ. Model. 9, 385–407 (a preliminary version was Impact Preliminary Working Paper IP-54. IMPACT Project, Melbourne, 1991).

Horridge, J.M., Parmenter, B.R., Pearson, K.R., 1993. ORANI-F: A general equilibrium model of the Australian economy. Econ. Financ. Comput. 3, 71–140.

Hudson, E.A., Jorgenson, D.W., 1974. US energy policy and economic growth, 1975–2000. Bell J. Econ. Manag. Sci. 5, 461–514.

Johansen, L., 1960. A Multisectoral Study of Economic Growth. Contributions to Economic Analysis 21. North-Holland, Amsterdam.

Kallrath, J., 2004. Modeling Languages in Mathematical Optimization. Kluwer, Boston, MA.

Meeraus, A., 1983. An algebraic approach to modeling. J. Econ. Dynam. Contr. 5, 81–108.

Pearson, K.R., 1988. Automating the computation of solutions of large economic models. Econ. Model. 5, 385–395.

Press, W.H., Teulosky, S.A., Vettering, W.T., Flannery, B.P., 1992. Numerical Recipes in C: The Art of Scientific Computing, second ed.). Cambridge University Press, Cambridge.

Rutherford, T.F., 1987. Applied General Equilibrium Modeling. PhD Dissertation. Department of Operations Research, Stanford University.

Rutherford, T.F., 1999. Sequential joint maximization, energy and environmental policy modeling. In: Weyant, J. (Ed.), International Series in Operations Research and Management Science 18. Kluwer, Boston, MA.

Scarf, H.E., 1967. On the Computation of Equilibrium Prices. Cowles Foundation Discussion Papers 232. Cowles Foundation for Research in Economics, Yale University, New Haven, CT.

Shoven, J.B., Whalley, J., 1972. A General Equilibrium Calculation of the Effects of Differential Taxation of Income from Capital in the US. Cowles Foundation Discussion Papers 328. Cowles Foundation for Research in Economics, Yale University, New Haven, CT. see also Journal of Public Economics, 1, 281–321 (1972).

Taylor, L., Black, S.L., 1974. Practical general equilibrium estimation of resources pulls under trade liberalization. J. Int. Econ. 4, 37–58.