Impact Project

Impact Research Centre, Baldwin Spencer Building (2nd floor)
The University of Melbourne, PARKVILLE, VIC. 3052 AUSTRALIA

Telephone: (03) 344 7417 (*from overseas:* 61 3 344 7417)
Telex: AA 35185 UNIMEL
Telegrams: UNIMELB, PARKVILLE
Fax: (03) 347 7539 (*from overseas:* 61 3 347 7539)

GENERAL-PURPOSE SOFTWARE FOR INTERTEMPORAL MODELLING

by

George Codsi
Impact Research Centre

K.R. Pearson
La Trobe University and Impact Research Centre

and

Peter J. Wilcoxen
Harvard University

Preliminary Working Paper No. IP-51 May 1991

ISSN 1031 9034 *ISBN 0 642 10121 3*

ABSTRACT

Intertemporal modelling is becoming increasingly important in general equilibrium policy analysis. Issues such as natural resource management, pollution control, investment, the promotion of technical change and the accumulation of foreign debt all involve explicit intertemporal behavior and thus require intertemporal models to address adequately. Applied researchers have been slow to adopt the intertemporal paradigm because it can impose formidable computational requirements. In this paper we address this problem by presenting flexible, efficient new software which can solve complex intertemporal models in a fraction of the time required by conventional approaches.

Our intertemporal software is an extension of GEMPACK, a suite of general-purpose computer programs developed to streamline the implementation and solution of a wide class of applied partial or general equilibrium models. This paper describes GEMPACK's new intertemporal modelling capabilities which make it straightforward for modellers to implement and solve their models. Because GEMPACK operates on models written in a syntax similar to ordinary algebra, it is easy for researchers to revise their models whenever necessary.

In summary, this new software removes the computational impediments to intertemporal modelling, thereby allowing modellers to concentrate on the economics of their models.

Table of Contents

GENERAL-PURPOSE SOFTWARE FOR INTERTEMPORAL MODELLING

by

George Codsi, K.R. Pearson and Peter J. Wilcoxen[1]

1. INTRODUCTION

Intertemporal (or dynamic) models are those having equations linking variables at different points in time. A very simple dynamic model is a single capital accumulation equation giving next year's capital stock as a function of this year's capital stock and net investment. More complex models can include equations derived from intertemporal optimization and may involve adaptive or rational expectations.[2] In all cases, the distinguishing feature of a dynamic model is that it contains at least one equation constraining how a variable evolves over time. Such intertemporal equations will be either differential or difference equations depending on whether the model is formulated in continuous or discrete time. In this paper we describe how to use a general-purpose economic modelling software package, GEMPACK,[3] to implement and solve a wide range of dynamic models. We focus particular attention on models formulated in continuous time, although the methods we discuss are equally applicable to *discrete time models.*

In Section 2 we show how continuous time models can be converted to discrete time models by replacing derivatives by finite differences. GEMPACK allows flexibility in specifying the number of time instants and the gaps between them. By varying

1. The authors are at the Impact Centre, La Trobe University and the Impact Centre, and Harvard University, respectively.
2. For a detailed discussion of the theory of dynamic modelling and of numerical methods used to solve dynamic models, see Wilcoxen (1989) or Chapter 5 of Dixon, *et al.* (forthcoming).
3. GEMPACK is a suite of modelling software which runs on PCs and most mainframes. It is described in Pearson (1988) and Codsi and Pearson (1988).

the points modelled in this way, it is possible to obtain solutions to this discrete problem which are as close as desired to the true solutions of the original continuous time model; this is discussed in Section 7.

The theory and equations of a model are communicated to GEMPACK simply by preparing a text file containing a linearised representation of the equations. The software is able to produce true solutions of nonlinear models from this linearised representation.[4] Sections 3 and 4 contain a description of the procedure for expressing intertemporal models in the syntax expected by GEMPACK. Section 5 describes how base case scenarios can be constructed. In Section 6 we illustrate our procedures with a stylized model of forestry.

One feature of the method described here is that the whole system of equations (including the intertemporal equations) is solved simultaneously. Hence the values of all variables at all time instants are found at once. (That is, they are not successively determined one time instant at a time.) In particular the method makes no real distinction between the intertemporal equations and the others; thus it works well for models with any number of intertemporal equations, and terminal conditions that must be satisfied in the future are easily accommodated. The sparse matrix methods used by GEMPACK make it feasible to solve the resulting system of equations even when the number of time instants is quite large.

4. See Pearson (1991). Moreover, this is accomplished entirely by GEMPACK without the user needing to write any model-specific programs or subroutines.

2. DISCRETE APPROXIMATIONS FOR CONTINUOUS TIME MODELS

For continuous time models, the first step in using GEMPACK is to replace any time derivatives with appropriate discrete approximations.[5] These approximations are known as finite-difference formulae and are constructed from Taylor series expansions. For example, an approximation for a first derivative might be constructed as follows. First, expand the function of interest, say f, about a particular time t for an adjacent time $t+h$:

$$f(t+h) = f(t) + f'(t)h + O(h^2) \tag{1}$$

where f' is the derivative of f, h is a small step in time and $O(h^2)$ represents the Taylor series terms of order h^2 and above. Rearranging (1) and dropping the higher-order terms shows that:

$$\frac{f(t+h)-f(t)}{h} \approx f'(t) \tag{2}$$

The left side of (2) is a finite difference approximation to the derivative of f evaluated at time t. Since it was constructed using current and future values of f, it is known as a forward difference.

To see how this works in practice, consider building a finite difference representation of the continuous time capital accumulation equation below, which links the capital stock, K, gross investment, I, and the depreciation rate, δ:

5. This step (all of Section 2) is unnecessary for discrete time models.

$$K'(t) = I(t) - \delta K(t) \qquad (3)$$

where the prime symbol (′) denotes a derivative with respect to time. Replacing K' with a finite difference equation of the form shown in (2) gives the following:

$$\frac{K(t+h)-K(t)}{h} = I(t) - \delta K(t) \qquad (4)$$

This can be rearranged to produce the expression below:

$$K(t+h) = hI(t) + (1-\delta h)K(t) \qquad (5)$$

When h is one year, (5) becomes the familiar discrete-time capital accumulation equation. For reasons which will become evident below, we will not assume that h is always one.

3. EXPRESSING THE MODEL IN A FORM SUITABLE FOR TABLO

The heart of GEMPACK is TABLO, a program which translates models from symbolic into numerical form. TABLO accepts input files containing a linearised[6] version of the model's equations written according to a syntax which is very close to that of ordinary algebra. Thus, once a discrete approximation has been chosen for the model, the next step is to linearise it. Once this has been done, the model can be expressed in TABLO using several features designed for dynamic modelling.[7] In

6. In this paper we use Johansen's percentage change linearisation (see Chapter 3 of Dixon *et al.* (forthcoming)), but other linearisations are also acceptable.
7. Dynamic modelling capabilities were first included with Release 4.2 of GEMPACK.

particular, TABLO now allows variables and coefficients to be indexed by time, so it is possible to write equations and formulae which involve variables at different points in time.

One of the simplest equations that might appear in an dynamic model is an expression which relates contemporaneous variables and which must hold at all points in time. For example, a Cobb-Douglas utility function generates the following demand equation for good X at time t:

$$X(t) = \frac{\alpha C(t)}{PX(t)} \tag{6}$$

where α is the exponent of X in the utility function, PX is the price of X, and C is the total value of consumption. Thus, demand for X at time t depends on the price of X at t and total consumption at t, but not on any variables at other times. The linearised version of (6) is the following:

$$x(t) = c(t) - px(t) \tag{7}$$

where we have used lower case letters to represent percentage changes in the corresponding uppercase variables. In TABLO, (7) might be written:

```
EQUATION demand (all,t,alltime) x(t) = c(t) - px(t);        (8)
```

where *demand* is the name of the equation. Variable x might be declared in the following fashion:

```
VARIABLE (all,t,alltime) x(t);
```

with c and px declared in a similar way.

Equation (8) and the declaration of x illustrate the first difference between static and dynamic models in TABLO: in dynamic models, variables are subscripted by an index of time, in this case t. One consequence of this is that a quantifier must now be included in the equation statement to describe the domain of t. In equation (8), for example, t is allowed to range over set *alltime*. Sets that are used as the domain of time indicies are called "intertemporal sets".

Equations involving addition or subtraction introduce an additional difference between static and dynamic models in TABLO: some of the coefficients might need to be subscripted by time. Consider the accounting relationship shown below between savings, $S(t)$, after-tax income, $Y(t)$, and consumption, $C(t)$:

$$S(t) = Y(t) - C(t) \tag{9}$$

When linearised, this equation has the form:

$$s(t) = B_1(t)\cdot y(t) - (1-B_1(t))\cdot c(t) \tag{10}$$

where $B_1(t)$ is a share given by:

$$B_1(t) = \frac{Y(t)}{S(t)} \tag{11}$$

In TABLO, (10) and (11) would be expressed as follows:

```
EQUATION eq2 (all,t,alltime)
         s(t) = B1(t)*y(t) - (1-B1(t))*c(t);
FORMULA (all,t,alltime) B1(t) = Y(t)/S(t);
```

where *B1* is B_1 written in a form suitable for TABLO and $Y(t)$ and $S(t)$ are base case data.

The second of the two expressions shown above describes how coefficient $B1(t)$ is to be calculated from base case data on $Y(t)$ and $S(t)$. It differs from its static counterpart because Y and S are subscripted by time. This brings out an important feature of dynamic models: the base case is an entire trajectory of the model over the period of interest, not just a single equilibrium at a particular point in time. Put another way, a dynamic model is linearised about an equilibrium trajectory, while a static model is linearised about a single equilibrium point.[8] Since the ratio of Y to S in the base case may change over time, the coefficient B_1 should change as well. Thus, the expression must be applied for each t in *alltime*.

Intertemporal models typically contain two types of equations. Firstly there are those relating variables at different points in time, such as (5) above, or containing a derivative with respect to time (which reflects changes as time varies), such as (3) above. Secondly there are those such as (7) and (10) above which only involve contemporaneous variables (and have no time derivatives). The first of these are the ones that distinguish dynamic models from others, and we refer to them as "inter-temporal" or "dynamic" equations. The second can perhaps most accurately be

8. In Section 5 we will discuss how a base case can be constructed for a dynamic model.

referred to as "intratemporal" equations; however, because of the similarity of this word to "intertemporal", we will also use "intraperiod" to describe these equations.

Like intraperiod equations, dynamic equations are quite easy to express in TABLO. Consider the discrete-time capital accumulation equation given below:

$$K(t+1) = I(t) + (1-\delta)K(t) \tag{12}$$

This equation is just (5) with h is set to one. Converting this to percentage change form produces the following:

$$k(t+1) = S_1(t)\cdot i(t) + (1-S_1(t))\cdot k(t) \tag{13}$$

where $S_1(t)$ is a base case share given by the expression below:

$$S_1(t) = \frac{I(t)}{K(t+1)} \tag{14}$$

In TABLO, (13) might be written:

```
EQUATION ka (all,t,fwdtime)
         k(t+1) = S1(t)*i(t) + (1-S1(t))*k(t);
```

where $S1(t)$ could be calculated using the expression:

```
FORMULA (all,t,fwdtime) S1(t) = I(t)/K(t+1);
```

In these two equations the domain of t is a set called *fwdtime*, rather than *alltime*. This new set contains each element of *alltime* whose successor is also in *alltime* – that is, all t for which $t+1 \in alltime$. In particular, *fwdtime* excludes the last point in *alltime* since, by construction, *alltime* does not include the successor to that point. We will discuss *alltime* and *fwdtime* in more detail in Section 4.

In translating dynamic models to TABLO, no particular difficulty is imposed by forward-looking equations. Consider the equation below which defines the change in human wealth over one year:

$$W'(t) = R(t)W(t) - Y(t) \tag{15}$$

where W is human wealth, Y is labour income and R is the interest rate. If we replace the left term by a finite difference approximation similar to (2) and set the step h equal to one, we obtain:

$$W(t+1) - W(t) = R(t) \cdot W(t) - Y(t) \tag{16}$$

which looks somewhat more familiar in the form:

$$W(t+1) = (1+R(t))W(t) - Y(t) \tag{17}$$

Equation (17) can be linearised and converted to TABLO format in precisely the manner used earlier for the capital accumulation equation.

One further type of equation arises in dynamic modelling because certain variables are governed by special equations at particular points in time. For example, the initial capital stock is usually known at the beginning of the simulation, so an equation like the following holds:

$$K(0) = K_0 \tag{18}$$

where K_0 is the known (and exogenous) initial value of the capital stock. Following the nomenclature of differential equations, expressions like (18) are called "boundary conditions". Dynamic models will have one boundary condition for each dynamic equation derived from a first-order differential or difference equation. If the original differential equations are of higher order, more boundary conditions will be required.

Boundary conditions can take several forms. In the simplest case, a boundary condition might give the initial value of some variable in the model; equation (18) is an example. In other circumstances, a more complicated equation may have to be used. For example, a boundary condition for the human wealth equation in (15) might be the following:

$$W(T) = \frac{Y(T)}{R(T)} \tag{19}$$

where T is the largest time in the interval of interest. (This condition would apply when T is large enough for the model to have essentially reached the steady state.) In linearised form, (19) becomes:

$$w(T) = y(T) - r(T) \tag{20}$$

This can be expressed in TABLO as:

```
EQUATION endcond (all,t,endtime) w(t) = y(t)-r(t);
```

where *endtime* is a set containing just one time instant, namely T.

Thus, dynamic models include three types of equation that are not present in static models: intraperiod equations, dynamic equations, and boundary conditions. Intraperiod equations are fairly similar to those appearing in static models, except that they hold at many points in time. Dynamic equations, on the other hand, are not similar to anything appearing in static models because they explicitly relate variables at different points in time. Finally, dynamic models also include boundary conditions. With these kinds of equation in mind, we now discuss the sets describing the domain of t in more detail.

4. ESTABLISHING THE DOMAIN OF THE TIME INDEX

Dynamic models capture behaviour over an interval of time. Without loss of generality, we may represent this interval by $[0,T]$ where 0 is an arbitrary starting date and T is some later date. Often T will be very large, sometimes more than a hundred years. To simulate a model over a long period of time it is necessary to divide up the original interval into a number of subintervals. The mathematical reason for this comes from the finite difference approximations introduced in Section 2, but the intuition behind it is fairly clear: if we want to know how the model behaves between 0 and T, we should expect to have to solve it at some intermediate points. This is the motivation for defining the model's equations over sets of points in time. In the remainder of this section we will discuss these sets in detail.

Replacing the model's time derivatives with finite difference formulae produces a system of equations that approximates the model near a particular point in time. Put another way, each finite difference equation is a local approximation to one of the original differential equations. These approximations are accurate to the order of the difference formulae; equation (5), for example, is accurate to $O(h)$. Thus, the model's numerical accuracy will depend heavily on the step size, h, used in constructing the difference formulae. Since the ending date, T, is typically very large, representing the entire interval with a single difference equation would usually produce a devastating amount of truncation error.

A concrete example makes this point clear. Suppose we were interested in modelling an economy over the next fifty years. If today's investment and capital are 1 and 5, respectively, and if the depreciation rate is 0.1, inserting those values into (5) and setting h to 50 suggests that fifty years from now, the capital stock should be 30. The true answer, obtained by integrating the original differential equation and assuming that investment is constant at 1 for the next fifty years, is very close to 10. Thus, using a single step to represent the entire interval produces a terrible estimate of the final capital stock. On the other hand, using today's investment and capital to compute the capital stock next year (so h is equal to 1) would produce a very good estimate: 5.50 when the true value is 5.48. Finite difference approximations are very useful over reasonably short intervals, but are not appropriate for long periods of time.

However, truncation error can be minimized by breaking the interval of interest up into a sequence of adjoining subintervals.[9] Then, each subinterval can be assigned

9. See Wilcoxen (1989) for a more complete discussion of how this subintervals can be used to minimize truncation error.

its own difference equation which will only hold over a fairly short period of time. This, in turn, reduces truncation error. For example, if the period of interest were $[0,T]$, two equal subintervals could be used: $[0,T/2]$ and $[T/2,T]$. Then, equation (5) would be replaced by the following:

$$K(T/2) = hI(0) + (1-\delta h)K(0) \qquad (21)$$

$$K(T) = hI(\frac{T}{2}) + (1-\delta h)K(\frac{T}{2}) \qquad (22)$$

Together, (21) and (22) approximate (5) over the interval $[0,T]$ with an order of accuracy $O(T/2)$. Thus, by using two finite difference equations, truncation error has been reduced by about fifty percent. If T is large, as it often is in dynamic models, using a step size of $T/2$ might still produce an unacceptable amount of truncation error. In that case, more subintervals would have to be used. If necessary, the solution can be made arbitrarily accurate by using a sufficiently large number of subintervals.

In general, N adjoining subintervals will be defined by $N+1$ time instants which we will call a "grid" of points. In the example above there are two subintervals and the corresponding grid is $\{0, T/2, T\}$. In most dynamic models the presence of intraperiod equations makes it necessary to be able to refer to each of the model's variables at each grid point. To facilitate this in TABLO, we can define a special set to represent the grid. This set can then be used to quantify declarations of variables, coefficients, formulae and equations, as *alltime* and *fwdtime* were in the examples above.

As we will discuss in Section 7, it can be very useful to keep the number of intervals (and thus points) in the grid flexible. TABLO's ability to handle sets whose size is defined at run time makes this particularly easy. A typical TABLO set declaration for a grid with N intervals (and thus $N+1$ grid points) is:

```
    SET (INTERTEMPORAL) alltime MAXIMUM SIZE 101 (p[0]-p[N]);    (23)
```

Several features of (23) need explanation. At the left, the set qualifier "*INTERTEMPORAL*" instructs TABLO that certain arithmetic operations are allowed on the set; in particular, that the expression $t+1$ (when t is an element of the set) is to be interpreted as the successor to element t. Moving to the right, the phrase "*MAXIMUM SIZE 101*" states that there will be up to 101 elements in *alltime*, although there may be fewer. At the far right, the expression "$(p[0]-p[N])$" is an abbreviation for the set of points $\{p0,p1,p2,...,pN\}$. N itself is an integer coefficient giving the number of desired time intervals. It would have been declared prior to (23), possibly with the statements below:

```
COEFFICIENT (INTEGER) N;
READ N FROM TERMINAL;
```

The second of these allows the user to specify the value of N interactively.

An important feature of (23) is that it declares the grid without actually assigning dates to its points. Instead it only gives the grid points symbolic names, such as *p0* and *p1*. This makes it easy to change the dates associated with the grid points, which is important in obtaining acceptable numerical accuracy. With *alltime* defined as shown in (23), a TABLO variable like $x(t)$ must be interpreted carefully. In particular, $x(t)$ is not the value of x at time t since t is only an index and is not a date in itself. Instead, $x(t)$ is the value of x at grid point number t. We will discuss how actual dates are associated with grid points shortly.

It is often useful to declare subsets of the grid. For example, the set *fwdtime* used in Section 3 is a subset of *alltime* consisting of the grid points in *alltime* whose successor is also in *alltime*. Thus, *fwdtime* is the domain of t where $t+1$ is in *alltime*. As a practical matter, *fwdtime* is the domain of t where it is possible to compute a forward difference. Because finite difference formulae often involve forward differences, *fwdtime* is a very important set. It could be declared using the following two statements:

```
SET (INTERTEMPORAL) fwdtime MAXIMUM SIZE 100 (p[0]-p[N-1]);
SUBSET fwdtime IS SUBSET OF alltime;
```

These statements create a set called *fwdtime* which consists of the first N elements of *alltime*, *p0* through *p[N-1]*. This illustrates the role of symbolic names in set declarations: they allow subsets to be expressed clearly and unambiguously.

A third set we find useful is the difference between *alltime* and *fwdtime*. Since this is just the last point in *alltime*, we could call it *endtime* and declare it as follows:

```
SET (INTERTEMPORAL) endtime SIZE 1 ( p[N] );
SUBSET endtime IS SUBSET OF alltime;
```

This set is used for handling boundary conditions that apply at the terminal time.

Actual dates can be associated with grid points by reading them in from a data file. Given a text file called "dates.dat" containing the desired dates, the following TABLO statements would create an appropriate coefficient called *date*:

```
FILE (TEXT) time "dates.dat";
COEFFICIENT (all,t,alltime) date(t);
READ date FROM FILE time;
```

This approach allows the dates associated with grid points to be read in from a data set in exactly the same way that other data are read. This makes the model much more flexible than it would be if specific dates were coded into the TABLO file directly. In particular, it facilitates changing the overall duration of simulations and it allows the length of grid intervals to be varied. The latter feature is an important means of obtaining accurate solutions at minimum cost.[10]

When dates are read from a file, the step size between grid points can be calculated within TABLO. For example, a coefficient *dt* could be defined as follows:

```
FORMULA (all,t,fwdtime) dt(t) = date(t+1) - date(t);
```

where *date(t)* is the date associated with grid point t. In this case, $dt(t)$ gives the time between grid point t and its successor $t+1$. In other words, $dt(t)$ gives the length of the next grid interval. This new entity can be used instead of h in the finite difference formula given in (2) to produce the expression below:

10. See Wilcoxen (1989).

$$\frac{f(t+1)-f(t)}{dt(t)} \approx f'(t) \tag{24}$$

where we have adopted the convention of WRITING F(T) TO REPRESENT F(DATE(T)) and f(t+1) to represent f(date(t+1)). We will continue using this notation for the remainder of the paper.

Using a grid of unevenly spaced points means that care must be exercised when dynamic equations are expressed in TABLO form. Consider the simple capital accumulation equation shown below:

$$K'(t) = I(t) - \delta K(t) \tag{25}$$

On an uneven grid, the finite difference version of this would be the following:

$$\frac{K(t+1)-K(t)}{dt(t)} \approx I(t) - \delta K(t) \tag{26}$$

Rewriting this slightly gives:

$$K(t+1) = dt(t)I(t) + (1-\delta dt(t))K(t) \tag{27}$$

The percentage change version of this can be expressed in TABLO as:

```
EQUATION cap k(t+1) = S1(t)*i(t) + (1-S1(t))*k(t);
```

where $S1(t)$ is a share defined as follows:

$$S1(t) = \frac{dt(t)I(t)}{K(t+1)} \qquad (28)$$

Notice that the grid interval length $dt(t)$ appears in this expression.

5. CONSTRUCTING A BASE CASE

GEMPACK requires one solution of the model (an initial equilibrium) as a starting point for its calculations. Other solutions are calculated and reported as perturbations from this original equilibrium. For a static model this initial equilibrium is a data set showing the values of all relevant variables at one point in time. For an intertemporal model, however, data is needed on the values of these variables at every time instant covered by the model. We refer to this large collection of data as a *base case;* it can be thought of as a collection of data bases (one for each grid point) which satisfies the (nonlinear) equations of the model - both the intraperiod and the dynamic equations. The coefficients of the linearised equations of the model are calculated from this base case data set.

Usually some of these grid points will be in the future, so it will be impossible to obtain the base case from historical data. Rather, it will have to be constructed as a solution to the model for a particular scenario of exogenous shocks. That is, a guess must be made about the future path of exogenous variables and then a corresponding solution to the model obtained. The difficulty of this depends on what behaviour is required of the base case exogenous variables.

The easiest base case to construct, and the one that has dominated intertemporal modelling to date, is a steady state. In this approach the base case values of future exogenous variables are assumed to be constants. Using these constants, the initial data set is adjusted so that the model will replicate itself from year to year as long as the exogenous variables remain at their original values. Thus, the base case consists of an arbitrary string of future points in time which look just like the initial period. Such a scenario is fairly easy to construct because it only requires obtaining a steady state solution to the model. We discuss an example in Section 6.

A different sort of base case is needed when future exogenous variables are assumed to be constant at their initial levels (as in the steady state case), but the model's state variables, such as the capital stock, are not initially at their steady state values. Intuitively, such a model will drift toward the steady state, approaching it asymptotically. For convenience, we will refer to this as a "disturbed-state" base case. Finally, a third type of base case is needed when the exogenous variables are not assumed to be constant in the future. This situation is routine when the model is to be used to generate counterfactual simulations during a historical period, so we will refer to this class as "historical" base cases.

Both disturbed-state and historical base cases require finding a full intertemporal solution to the model. This appears to present a problem for GEMPACK because it is usually only used to solve for perturbations around a known solution. However GEMPACK can be used to find a base case by either of the two procedures described below. The first involves solving a static version of the model to find a steady state solution to the intertemporal model and then returning the exogenous variables to their desired values. The second, which was suggested by our colleague Mark Horridge, involves adding slack variables to the intertemporal equations so that the initial data satisfies the modified equations. We have used this method with an intertemporal version of ORANI and believe it may prove easier to use in general than the first method.

(1) First, a set of changes in exogenous variables is found which would make the initial equilibrium a steady state. This step, in other words, finds exogenous variables for which the observed economy data would be a steady state. This can be accomplished by solving a static version of the model in which all derivatives are set to zero and the corresponding exogenous variables are endogenized. Once the steady state has been obtained, it can be used as the base case for an intertemporal simulation in which the exogenous variables are returned to their true values. The solution to this experiment will be the true base case of the model.

(2) Begin with a data base which is known to be a solution of the intraperiod equations of the model at one grid point. (This is just a data set for the underlying static model.) Now consider the full data set obtained by replicating this one data set to all grid points. This will satisfy all intraperiod equations but usually not the intertemporal ones. To overcome this problem, add a slack variable to each intertemporal equation. For example, the capital accumulation equation (12) above could be written:

$$K(t+1)= \Big(I(t) + (1-\delta)K(t) \Big) F(t)$$

where $F(t)$ is the new slack variable. The full data set is a solution of the modified model (consisting of the intraperiod equations and the modified intertemporal ones) since values of the slack variables satisfying the modified equations can easily be calculated. Of course we need a solution of the original (unmodified) equations. To obtain this, simply carry out a simulations with the modified model in which the slack variables (which are naturally exogenous in the modified model) are forced to their desired values. (For example, $F(t)$ above would be forced to 1.) The new solution of the modified model is the desired solution of the original equations. In carrying out this simulation with the modified model, other exogenous variables of the model can also be given shocks to bring them to the desired values if a historical base case is required.

Of course, with either method it will be important to know that the base case produced is an accurate solution to the original nonlinear equations of the model. Accordingly, all simulations referred to above should be carried out using the multistep features of GEMPACK in order to avoid linearisation errors.

6. EXAMPLE: A SIMPLE MODEL OF FORESTRY

We now present a simple intertemporal model of forestry to provide a concrete illustration of the principles discussed so far. In order to focus attention on the most important aspects of building the model in GEMPACK, we have deliberately based this model on a highly stylized description of forestry. This is for exposition only; a more sophisticated and realistic model could easily be built.

Consider a firm faced with the following optimization problem. It owns a plot of land on which G new, fully-grown trees appear each year. The growth rate G is exogenous and does not depend on the firm's behaviour, nor on the number of trees already present. If not cut down, the trees accumulate into a stock $S(t)$ at time t. However, the firm can fell trees and sell them for price $P(t)$ at time t. Suppose the firm fells trees at a rate $F(t)$ at time t and the rate of expenditure on tree felling is $C(S,F)$. Then the firm earns short run profit at the rate $\pi(t)$ given by:

$$\pi(t) = P(t)F(t) - C(S(t),F(t)) \tag{29}$$

Finally, suppose the firm wants to choose the path of future felling rates in order to maximize the present value of its short run profits. Thus, the firm solves the following optimal control problem:

$$\max \int_0^\infty (PF - C(S,F)) e^{-rt} \, dt \tag{30}$$

$$\text{subject to } S' = G - F \tag{31}$$

where r is the interest rate, assumed to be exogenous and constant over time.

The first order conditions for this problem can be shown to be the following:[11]

$$\lambda = P - C_F \tag{32}$$

$$\lambda' = r\lambda + C_S \tag{33}$$

$$S' = G - F \tag{34}$$

where λ is a multiplier, the prime symbol (′) denotes a derivative with respect to time, and C_F and C_S represent $\partial C/\partial F$ and $\partial C/\partial S$, respectively.

11. See Wilcoxen (1989) or Kamien and Schwartz (1981) for a discussion of how these conditions are derived.

The remaining step in formulating the model is to choose a functional form for C. For simplicity, we assume that costs are proportional to the wage rate, to the rate of tree felling and to the ratio of the felling rate to the total stock. If the wage rate is given by W, the cost function is then:

$$C = \theta F W \left(\frac{F}{S}\right) \tag{35}$$

where θ is a parameter. Using this expression, it is straightforward to show from (32) that the optimal felling rate is given by the expression below:

$$F(t) = \frac{(P(t)-\lambda(t))S(t)}{2\theta W(t)} \tag{36}$$

Inserting this into the first order conditions (33) and (34) above gives the model's equations of motion:

$$\lambda' = r\lambda - \frac{(P-\lambda)^2}{4\theta W} \tag{37}$$

$$S' = G - \frac{(P-\lambda)S}{2\theta W} \tag{38}$$

These equations summarize the behaviour of our forestry firm. To solve them numerically we next construct an appropriate finite difference approximation. Recall the forward difference expression in equation (24):

$$\frac{f(t+1)-f(t)}{dt(t)} \approx f'(t) \tag{39}$$

Applying this expression to the model's equations of motion produces two finite difference equations:

$$\frac{\lambda(t+1)-\lambda(t)}{dt(t)} = r\lambda(t) - \frac{(P(t)-\lambda(t))^2}{4\theta W(t)} \tag{40}$$

$$\frac{S(t+1)-S(t)}{dt(t)} = G - \frac{(P(t)-\lambda(t))\cdot S(t)}{2\theta W(t)} \tag{41}$$

These expressions can easily be converted to percentage change form. Using (40) as an example, the first step is to rearrange it slightly to obtain:

$$\lambda(t+1) = (1+r\cdot dt(t))\lambda(t) - dt(t)\,\frac{(P(t)-\lambda(t))^2}{4\theta W(t)} \tag{42}$$

Translating (42) into percentage change form gives the expression below:[12]

$$\lambda^*(t+1) = S_1(t)\lambda^*(t) + S_2(t)(\,2\cdot S_3(t)p(t)+2\cdot S_4(t)\lambda^*(t)-w(t)\,) \tag{43}$$

where λ^* is the percentage change in λ and coefficients S_1-S_4 have the following definitions:

$$S_1(t) = \frac{(1+r\cdot dt(t))\cdot\lambda(t)}{\lambda(t+1)} \tag{44}$$

$$S_2(t) = 1-S_1(t) \tag{45}$$

$$S_3(t) = \frac{P(t)}{P(t)-\lambda(t)} \tag{46}$$

12. See Impact Project (1988) or Chapter 3 of Dixon, *et al.* (forthcoming) for details on how equations are converted into percentage change form.

$$S_4(t) = 1 - S_4(t) \tag{47}$$

To complete the forestry model we need to add two boundary conditions to the equations of motion. The first of these is very intuitive: we require that the solution begin at the original stock of trees. The second condition we impose is that model must arrive at its steady state by the terminal date of the simulation.[13] This can be implemented by requiring that $\lambda' = 0$ at T. Thus, the boundary conditions add the following two equations to the model:

$$S(0) = S_0 \tag{48}$$

$$\lambda'(T) = 0 \tag{49}$$

where S_0 is the original stock of trees. Equation (48) is trivial to convert to percentage change form and can be implemented simply by making the percentage change in the initial stock of trees exogenous. Equation (49) can best be implemented by inserting it into (37) and converting the result into percentage change form to give:

$$\lambda^*(T) = -w(T) + 2S_3(T)p(T) + 2S_4(T)\lambda^*(T) \tag{50}$$

We assumed the interest rate is constant, so it does not appear in (50).

This leads naturally to an important point about steady state base cases for the model. In steady state solutions, all derivatives are zero, so from (37) and (38), we

13. See Wilcoxen (1989) for a more detailed explanation of boundary conditions and an argument justifying this particular choice.

have:

$$0 = r\lambda - \frac{(P-\lambda)^2}{4\theta W} \tag{51}$$

$$0 = G - \frac{(P-\lambda)S}{2\theta W} \tag{52}$$

Calculating a steady state base case for the model means finding values of F, P, W, G, λ, and S satisfying (51), (52) and (36) given r and θ. We can choose three of these independently and use (51), (52) and (36) to compute the others. The most natural exogenous variables are P, W and G, so for our sample model we have set:

$$r=0.05, \quad \theta=10, \quad P=24, \quad W=1, \quad \text{and} \quad G=100 \tag{53}$$

which gives:

$$\lambda=18, \quad S=1000/3, \quad \text{and} \quad F=100 \tag{54}$$

These values are a steady state and can be used as the base case of the model.

To formulate the model as a TABLO Input file, we follow the procedure described in Section 4.[14] First we define the model's intertemporal sets. We choose to keep the number of grid intervals flexible, as advocated in Section 4, so we allow for N grid points. Thus, we begin the file with the declaration of N, followed by the

14. The full TABLO Input file for this model is shown in Appendix A. In this section we discuss several of the file's most important features.

statement that its value will be read from the terminal. Following that, *alltime*, *fwdtime* and *endtime* are defined in terms of N. The relevant statements (see Section 4) are as follows:

```
COEFFICIENT (INTEGER) N;
READ N FROM TERMINAL;

SET (INTERTEMPORAL) alltime # all time periods #
    MAXIMUM SIZE 101 ( p[0] - p[N] );
SET (INTERTEMPORAL) fwdtime # domain of fwd diffs #
    MAXIMUM SIZE 100 ( p[0] - p[N - 1] );
SET (INTERTEMPORAL) endtime # ending time #
    SIZE 1  ( p[N] );

SUBSET fwdtime IS SUBSET OF alltime;
SUBSET endtime IS SUBSET OF alltime;
```

The next step is to associate specific dates with each of the grid points. If these dates are stored on a text file they could be read using the TABLO instructions below:

```
FILE (TEXT) time;
COEFFICIENT (all,t,alltime) year(t);
READ year FROM FILE time;
```

We can calculate the difference between consecutive dates (which will be used in finite difference approximations where $dt(t)$ appears), as shown:

```
COEFFICIENT (all,t,fwdtime) dt(t);
FORMULA (all,t,fwdtime) dt(t) = year(t+1) - year(t);
```

These differences can only be computed when both $year(t)$ and $year(t+1)$ are known, so they are only defined on the subset *fwdtime*, not on *alltime*. An example of file *time* is shown later in this section.

Next we declare the model's variables. These are all interpreted as percentage changes from the base case.

```
VARIABLE (all,t,alltime) p(t)   # price of trees          #;
VARIABLE (all,t,alltime) w(t)   # wage rate               #;
VARIABLE (all,t,alltime) lam(t) # shadow value of capital #;
VARIABLE (all,t,alltime) s(t)   # stock of trees          #;
VARIABLE (all,t,alltime) g(t)   # growth rate             #;
VARIABLE (all,t,alltime) f(t)   # felling rate            #;
```

Following this we give instructions which establish the base case. Ordinarily, we would expect to need to know the values of the parameters r and θ and the base case values of all 6 variables (P,W,G,F,LAM, and S) at each grid point. To distinguish these base case values from the variables with the same names (but different interpretations) we add "$\_B$", for "base case", to each name. For this model, it turns out that we only need the base case values G_B, P_B, LAM_B and S_B and that we do not need to know the value of θ (although its value is needed to calculate the base values). We declare the parameter r and the 4 coefficients, declare a text file *basedata* to hold the base case values of these, and give instructions to read the required base case values. The TABLO statements are shown below:

```
COEFFICIENT r ; FORMULA r = 0.05 ;
COEFFICIENT (all,t,alltime) G_B(t)    # Base values of G(t)      #;
COEFFICIENT (all,t,alltime) P_B(t)    # Base values of P(t)      #;
COEFFICIENT (all,t,alltime) LAM_B(t) # Base values of LAMBDA(t) #;
COEFFICIENT (all,t,alltime) S_B(t)    # Base values of S(t)      #;
FILE (TEXT) basedata ;
READ G_B   FROM FILE basedata ;
READ P_B   FROM FILE basedata ;
READ LAM_B FROM FILE basedata ;
READ S_B   FROM FILE basedata ;
```

Now we come to the equations. For each equation we must declare the coefficients occurring in it, give formulae for calculating their values from the base case values, and finally write the equation itself. For equation (43) – the percentage change form of (33) – we have the following:

```
COEFFICIENT (all,t,fwdtime) S1(t);
COEFFICIENT (all,t,fwdtime) S2(t);
COEFFICIENT (all,t,alltime) S3(t);
COEFFICIENT (all,t,alltime) S4(t);

FORMULA (all,t,fwdtime) S1(t)=[1+dt(t)*r]*LAM_B(t)/LAM_B(t+1);
FORMULA (all,t,fwdtime) S2(t)=1-S1(t);
FORMULA (all,t,alltime) S3(t)=P_B(t)/[P_B(t)-LAM_B(t)];
FORMULA (all,t,alltime) S4(t)=1-S3(t);

EQUATION d1 (all,t,fwdtime) lam(t+1)=S1(t)*lam(t) +
        S2(t)*( -w(t)+2*(S3(t)*p(t)+S4(t)*lam(t)) );
```

Notice that $S1(t)$ and $S2(t)$ are only defined over *fwdtime* since they relate to forward differences, while $S3(t)$ and $S4(t)$ are defined over *alltime*. Similar TABLO statements are required for the other equations and for the steady state boundary condition.

Finally, in order to calculate multi-step solutions accurately we need to include instructions on how the base data is updated after a simulation. This eliminates approximation errors that would be present if we calculated just the Johansen (1-step) solution. To update the four base case coefficients we add the following statements to the TABLO Input file:

```
UPDATE (all,t,alltime) G_B(t)   = g(t)   ;
UPDATE (all,t,alltime) P_B(t)   = p(t)   ;
UPDATE (all,t,alltime) LAM_B(t) = lam(t) ;
UPDATE (all,t,alltime) S_B(t)   = s(t)   ;
```

Each rule shows how the percentage change results on the right hand side are to be used to update the base case variable on the left. For example, the first rule states that the updated value of $G\_B(t)$ is equal to the old value of $G\_B(t)$ multiplied by $(1+g(t)/100)$, where $g(t)$ is the percentage change in G at time t. Each of the subsequent rules has a similar meaning.

To complete the model, the remaining step is to prepare the data files. First consider file *time*, which gives the dates associated with the model's grid points. Suppose we are interested in simulating an 80 year period using 20 grid intervals, all of the same length. In this case, N would be 20 and the years could be taken as {0,4,8,...,80}.[15] File *time* would then contain the following two lines:

```
1 20 (year values)
0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80
```

15. The years could just as easily have been defined as {1990,1994,1998,...,2070}, or according to any desired pattern.

The first line indicates that the data being described (in this case *year*) is a one-dimensional array of length 20. The second line contains the actual data.

The other file we must provide is *basedata*, which contains the base case values of G_B, P_B, LAM_B and S_B. If we use the steady state values given in (53) and (54) above, this would contain 2 lines for each coefficient, as shown below ("21*100" is an abbreviation understood by GEMPACK to mean 100 written 21 times):

```
1 21    (G_B values)
21*100
1 21    (P_B)
21*24
1 21    (LAM_B)
21*18
1 21    (S_B)
21*333.3333
```

Together, the TABLO file in Appendix A and the two data files just presented form a complete implementation of the forestry model. We conclude this section with a brief remark on how results from the model might be interpreted. Suppose we run a simulation using the grid and base case values given above and obtain values of 10, 6.3 and -5 for the first three components of the variable $f(t)$ (the percentage change in the felling rate). These are percentage changes from the base case, which is 100 at each grid point. Thus these results mean that the felling rate increases from 100 to 110 at year 0, increases from 100 to 106.3 at year 4, and decreases from 100 to 95 at year 8.

7. NUMERICAL ACCURACY

The numerical accuracy of finite difference formulae is discussed in detail in Wilcoxen (1989), but a few further remarks are in order here. In particular, it is easy to estimate the numerical accuracy of models implemented in GEMPACK, and to increase that accuracy when necessary. The truncation error introduced by finite difference formulae can be controlled to whatever degree is required.

Assessing the accuracy of a particular solution can be accomplished by simulating the experiment a second time using a grid that is twice as dense as the original. Since truncation error is $O(h)$ (when forward differences are used), the error at each point in the solution should decrease by about fifty percent on the double-density grid. Thus, the difference in the results between the two simulations gives a fairly good measure of the error.

If the error is unacceptably high, any of the measures described in Wilcoxen (1989) may be taken: the grid density can be increased uniformly, grid spacing may be rearranged, or different finite difference formulae may be used which are accurate to higher order. In addition, several simulations at different grid densities can be combined using Richardson's extrapolation to construct a solution that is more accurate than any of them in isolation.[16]

Finally, it is important to consider how the number of grid points, N, affects the computer time needed to solve a particular model. The overall size of an intertemporal model (the numbers of equations and variables) is proportional to the number N of grid points. If an existing static computable general equilibrium (CGE) model is

16. Richardson's extrapolation is described in Birkhoff and Rota (1978).

converted to an intertemporal model, the number of equations and variables in the intertemporal version will be approximately the number in the static version multiplied by N (since each variable and most equations in the static model get an extra t subscript).

This makes it clear that the computational costs of working with an intertemporal model are considerably greater than those for the associated static CGE model. Two parts of these costs need to be distinguished. Firstly, it is necessary to convert the symbolic equations to numerical form. This involves calculating all the shares from the base data, and the cost is roughly proportional to N. Secondly, there is the cost of doing an LU decomposition (see Duff (1977)) of a matrix of size $K \times K$ where K is the number of endogenous variables. Usually this cost is proportional to K^3, so (since K is proportional to N) if there are 10 grid points this cost would be multiplied by 1000 from that of the static model. Such a cost increase would be an enormous problem.

However, our initial empirical experience is that the LU decomposition cost only increases approximately in proportion to N rather than N^3. This is because most of the equations in an intertemporal CGE model are intraperiod equations. The preliminary reordering of the equations and variables done by the Harwell sparse matrix routine MA28 (see Duff (1977)) used by GEMPACK automatically detects the associated pattern in the matrix and is then able to break the whole problem into approximately N subproblems each of size $K \times K$. This efficiency feature of MA28 is thus very important for keeping the costs of intertemporal model solving within manageable bounds. Our conclusion is that to a first order of approximation, total computing costs are proportional to the number of grid points.

Finally, also note that the condensation facilities in GEMPACK can be used to reduce the size of an intertemporal model actually solved. This is another way of keeping computing costs within bounds.

8. CONCLUSION

In summary, the extended version of TABLO allows a wide variety of dynamic models to be formulated and solved within GEMPACK. This includes, but is not limited to, the perfect foresight models of intertemporal optimization currently popular in macroeconomics. Numerical accuracy can be measured and controlled to any extent desired. The most difficult step in implementing a model is constructing the base case. However, there are straightforward methods available to create base cases of arbitrary complexity. Thus, the extended version of TABLO provides a versatile and convenient method of building dynamic models.

9. REFERENCES

Birkhoff, Garrett and Gian-Carlo Rota (1978), *Ordinary Differential Equations*, New York: John Wiley and Sons.

Codsi, G. and K.R. Pearson (1988), "GEMPACK: General-Purpose Software for Applied General Equilibrium and Other Economic Modellers", Computer Science in Economics and Management, Vol. 1, pp 189-207.

Dixon, Peter B., B.R. Parmenter, Alan A. Powell and Peter J. Wilcoxen (forthcoming), *Notes and Problems in Applied General Equilibrium Economics*, Amsterdam: North-Holland.

Duff, I.S. (1977), "MA28 - A Set of FORTRAN Routines for Sparse Unsymmetric Linear Equations", Harwell Report R.8730 (HMSO, London), pp.104.

Impact Project (1988), "Stylized Johansen - An Illustrative CGE Model", Impact Project GEMPACK Document 23, University of Melbourne.

Pearson, K.R. (1988), "Automating the Computation of Solutions of Large Economic Models", Economic Modelling, Vol. 5, pp 385-395.

Pearson, K.R. (1991), "Solving Nonlinear Economic Models Accurately via a Linearized Representation", IMPACT Project Preliminary Working Paper (in preparation).

Wilcoxen, Peter J. (1989), "Intertemporal Optimization in General Equilibrium: A Practical Introduction," IMPACT Project Preliminary Working Paper No. IP-45, University of Melbourne.

A. Appendix: A TABLO File for the Forestry Model

In this appendix we present a complete TABLO Input file implementing the forestry model discussed in Section 6.

```
!---------------------------------------------------------------------!
!  TREES.T 1.52                                                        !
!  23 Nov 89 (PJW) and 01 Mar 91 (KRP)                                 !
!                                                                      !
!  A simple intertemporal model of forestry as described in           !
!   "General-Purpose Software for Intertemporal Modelling"            !
!    by George Codsi, K.R. Pearson and Peter J. Wilcoxen              !
!                                                                      !
!   (Start from an arbitrary base case)                                !
!   (Updates included to allow multi-step simulations)                 !
!---------------------------------------------------------------------!

!---------------------------------------------------------------------!
!  Number of grid intervals                                            !
!---------------------------------------------------------------------!

COEFFICIENT (INTEGER) N;
READ N FROM TERMINAL;

!---------------------------------------------------------------------!
!  Sets for describing periods                                         !
!---------------------------------------------------------------------!

SET (INTERTEMPORAL) alltime # all time periods # MAXIMUM SIZE 101
  ( p[0] - p[N] );
SET (INTERTEMPORAL) fwdtime # domain of fwd diffs# MAXIMUM SIZE 100
  ( p[0] - p[N - 1] );
SET (INTERTEMPORAL) endtime # ending time # SIZE 1  ( p[N] ) ;

SUBSET fwdtime IS SUBSET OF alltime;
SUBSET endtime IS SUBSET OF alltime;


!---------------------------------------------------------------------!
!  Variables                                                           !
!---------------------------------------------------------------------!
VARIABLE (all,t,alltime) p(t)     # Price of trees                 #;
VARIABLE (all,t,alltime) w(t)     # wage rate                      #;
VARIABLE (all,t,alltime) lam(t)   # shadow value of capital        #;
VARIABLE (all,t,alltime) s(t)     # stock of trees                 #;
VARIABLE (all,t,alltime) g(t)     # growth rate                    #;
VARIABLE (all,t,alltime) f(t)     # felling rate                   #;
```

```
!-----------------------------------------------------------------------!
!  Parameters                                                           !
!-----------------------------------------------------------------------!
COEFFICIENT r     ; FORMULA r     = 0.05 ;
! Theta is not needed here !


!-----------------------------------------------------------------------!
!  Base case                                                            !
!-----------------------------------------------------------------------!
COEFFICIENT (all,t,alltime) G_B(t)     # Base values of G(t)       #;
COEFFICIENT (all,t,alltime) P_B(t)     # Base values of P(t)       #;
COEFFICIENT (all,t,alltime) LAM_B(t)   # Base values of LAMBDA(t)  #;
COEFFICIENT (all,t,alltime) S_B(t)     # Base values of S(t)       #;
FILE (TEXT) basedata ;
READ G_B     FROM FILE basedata ;
READ P_B     FROM FILE basedata ;
READ LAM_B   FROM FILE basedata ;
READ S_B     FROM FILE basedata ;

!-----------------------------------------------------------------------!
!  Updates                                                              !
!-----------------------------------------------------------------------!
UPDATE (all,t,alltime) G_B(t)   = g(t)   ;
UPDATE (all,t,alltime) P_B(t)   = p(t)   ;
UPDATE (all,t,alltime) LAM_B(t) = lam(t) ;
UPDATE (all,t,alltime) S_B(t)   = s(t)   ;


!-----------------------------------------------------------------------!
!  Time                                                                 !
!-----------------------------------------------------------------------!
FILE (TEXT) time ;
COEFFICIENT (all,t,alltime) year(t);
READ year FROM FILE time ;

COEFFICIENT (all,t,fwdtime) dt(t);
FORMULA (all,t,fwdtime) dt(t) = year(t+1) - year(t);


!-----------------------------------------------------------------------!
!  Dynamic equations relating adjacent periods                          !
!-----------------------------------------------------------------------!
COEFFICIENT (all,t,fwdtime) S1(t);
COEFFICIENT (all,t,fwdtime) S2(t);
COEFFICIENT (all,t,alltime) S3(t);
COEFFICIENT (all,t,alltime) S4(t);

! The following are text equations (44-47) !
FORMULA (all,t,fwdtime) S1(t) = [1+dt(t)*r]*LAM_B(t)/LAM_B(t+1);
FORMULA (all,t,fwdtime) S2(t) =  1-S1(t);
FORMULA (all,t,alltime) S3(t) =  P_B(t)/[P_B(t)-LAM_B(t)];
FORMULA (all,t,alltime) S4(t) =  1-S3(t);
```

```
EQUATION d1 ! text equation (43) ! (all,t,fwdtime)
   lam(t+1) = S1(t)*lam(t)+S2(t)*( -w(t)+2*(S3(t)*p(t)+S4(t)*lam(t)) );


COEFFICIENT (all,t,fwdtime) S5(t);
COEFFICIENT (all,t,fwdtime) S6(t);
COEFFICIENT (all,t,fwdtime) S7(t);

FORMULA (all,t,fwdtime) S5(t) = S_B(t)/S_B(t+1);
FORMULA (all,t,fwdtime) S6(t) = dt(t)*G_B(t)/S_B(t);
FORMULA (all,t,fwdtime) S7(t) = 1-S5(t)-S6(t);

EQUATION d2 ! text equation (41) linearised ! (all,t,fwdtime)
   s(t+1) = S5(t)*s(t) + S6(t)*g(t)
          + S7(t)*( s(t) + S3(t)*p(t) + S4(t)*lam(t) - w(t) );


!----------------------------------------------------------------------!
!  Boundary conditions                                                 !
!                                                                      !
!  1. Initial stock condition imposed by setting s(0) exogenously.     !
!     This allows simulations to be run in which the initial forest    !
!     stock changes.                                                   !
!                                                                      !
!  2. Equation d1 [text equation (37) with lam constant]               !
!----------------------------------------------------------------------!
EQUATION b2 (all,t,endtime)
   lam(t) = -w(t)+2*( S3(t)*p(t)+S4(t)*lam(t) );

!----------------------------------------------------------------------!
!  Intraperiod equation                                                !
!                                                                      !
!     This is a linearised version of text equation (36)               !
!----------------------------------------------------------------------!
EQUATION e1 (all,t,alltime)
   f(t) + w(t) = s(t) + S3(t)*p(t) + S4(t)*lam(t) ;
```