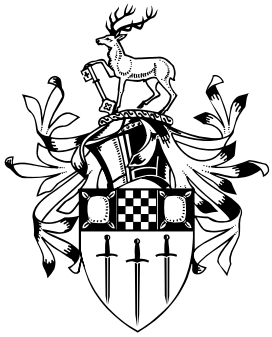
UniS

University of Surrey



Discussion Papers in Economics

TERMINAL CONDITIONS IN FORWARD-LOOKING ECONOMIC MODELS

By

Richard G. Pierse
(University of Surrey)

DP 10/06

Department of Economics
University of Surrey
Guildford
Surrey GU2 7XH, UK
Telephone +44 (0)1483 689380
Facsimile +44 (0)1483 689548
Web www.econ.surrey.ac.uk
ISSN: 1749-5075

Terminal conditions in forward-looking economic models

Richard G. Pierse
Department of Economics
University of Surrey, Guildford GU2 7XH, U.K.

February 24 2006

Abstract

In this paper we show how the popular L-B-J algorithm for solving forward-looking economic models using Newton methods can be generalised to allow for a block of terminal equations for variables that appear with a lead. The effect of choosing different types of terminal condition is explored in a simple stochastic growth model using WinSolve, a general nonlinear model solution package.

1 Introduction

The L-B-J algorithm, due to Laffargue (1990), Boucekkine (1995) and Julliard (1996), has become a popular method for solving non-linear economic models involving forward-looking variables. An application of Newton's method to the model equations stacked over time, the algorithm takes advantage of the special sparse structure of the Jacobian matrix to solve the linear Newton step equations efficiently, without having to invert or even store the complete matrix. The algorithm has been implemented in computer packages such as *Dynare* (Juillard, 1996), *Troll* (Hollinger, 1996) and *WinSolve* (Pierse, 2002), either using numerical derivatives (*Dynare*) or automatic derivatives (*Troll* and *WinSolve*). Juillard *et al.* (1998) compare Newton methods with alternative first-order methods such as Fair-Taylor (Fair and Taylor (1983)), using a variety of different models, and find that Newton methods are faster and more robust than first-order techniques.

One important issue with the solution of models involving forward-looking variables is the determination of terminal values. Any forward-looking model requires terminal values to be specified for all model variables that appear with a lead. In the standard L-B-J algorithm, the only possibility is to set terminal values to constant exogenous values, since these do not affect the Newton method. However, it is more usual to want to impose terminal conditions that either force terminal values to revert to deterministic steady state values or make them follow a simple rule such as constant level or constant growth at the terminal date. To do this, the L-B-J algorithm needs to be extended to incorporate an extra block of equations defining the terminal conditions. An algorithm for this is presented in this paper and this has been implemented in *WinSolve*.

The rest of the paper is structured as follows: Section 2 briefly outlines the L-B-J algorithm for solving forward-looking models and Section 3 shows how the algorithm can be extended by augmenting the Newton equations to be solved with a block of terminal condition equations for the forward-looking variables. In Section 4, a simple stochastic growth model is described and in Section 5 the results of stochastic simulations with this model are compared when different types of terminal condition are imposed on the forward-looking variable, all simulations conducted using the software package *WinSolve*[1]. The final section summarises and presents some conclusions.

2 The L-B-J algorithm

The general nonlinear deterministic forward-looking model is defined by the equations

$$\mathbf{f}_t(\mathbf{y}_t, \mathbf{y}_{t+1}, \cdots, \mathbf{y}_{t+q}, \mathbf{y}_{t-1}, \cdots, \mathbf{y}_{t-p}, \mathbf{x}_t; \boldsymbol{\theta}) = \mathbf{0} \quad , \quad t = 1, \cdots, T \qquad (1)$$

where $\mathbf{y}_t$ is an $n \times 1$ vector of *endogenous* variables in time period t, $\mathbf{x}_t$ is an $m \times 1$ vector of current and lagged exogenous variables, $\mathbf{f}_t$ is an $n \times 1$ vector valued function and $\boldsymbol{\theta}$ is a vector of parameters, p is the longest lag in the model and q is the longest lead. This system represents a set of n nonlinear equations over T time periods. Stacking the equations over all time periods produces a set of nT equations. The Jacobian matrix of this stacked system

[1] A free trial version is available for download from the Internet at web address www.econ.surrey.ac.uk/winsolve.

has a special structure and looks like

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_1 & \mathbf{F}_1^1 & \cdots & \mathbf{F}_1^q & & & & \\ \mathbf{B}_2^1 & \mathbf{J}_2 & \mathbf{F}_2^1 & \cdots & \mathbf{F}_2^q & & & \\ \vdots & \ddots & \ddots & \ddots & \cdots & \ddots & & \\ \mathbf{B}_p^p & \cdots & \mathbf{B}_p^1 & \mathbf{J}_p & \ddots & \cdots & \ddots & \\ & \ddots & \cdots & \ddots & \ddots & \ddots & \cdots & \mathbf{F}_{T-k}^q \\ & & \ddots & \cdots & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & \cdots & \ddots & \mathbf{J}_{T-1} & \mathbf{F}_{T-1}^1 \\ & & & & \mathbf{B}_T^p & \cdots & \mathbf{B}_T^1 & \mathbf{J}_T \end{bmatrix} \tag{2}$$

where

$$\mathbf{J}_t = \frac{\partial \mathbf{f}_t}{\partial \mathbf{y}_t'} \quad , \quad \mathbf{F}_t^i = \frac{\partial \mathbf{f}_t}{\partial \mathbf{y}_{t+i}'} \quad , \quad \mathbf{B}_t^i = \frac{\partial \mathbf{f}_t}{\partial \mathbf{y}_{t-i}'}$$

are all matrices of dimension $n \times n$.

The *Stacked Newton method* applies Newton's method (Newton (1686)) to the stacked system. This involves iterating on the set of nT equations

$$\mathbf{J}(\mathbf{y}^s - \mathbf{y}^{s-1}) = -\mathbf{f}(\mathbf{y}^{s-1}) \tag{3}$$

where $\mathbf{y}^s$ is the $nT \times 1$ vector of stacked values of the endogenous variables in iteration s and $\mathbf{f}$ is the $nT \times 1$ vector valued function formed by stacking $\mathbf{f}_1 \cdots \mathbf{f}_T$. Iterations start from an initial guess at the solution, $\mathbf{y}^0$, and terminate when a convergence criterion such as

$$\max_j \left| \frac{\mathbf{y}_j^s - \mathbf{y}_j^{s-1}}{\mathbf{y}_j^{s-1}} \right| < \varepsilon$$

has been satisfied, for some small value of ε.

Each iteration of Newton's method involves the solution of a set of nT equations. When either n or T is big, the matrix will be large and this causes two problems. Firstly, storing the complete matrix consumes a lot of computer memory. Secondly, using standard solution techniques, the cost of solving a set of equations is roughly cubic in the order of the matrix and this cost will quickly become prohibitive. However, the structure of the Jacobian matrix (2) is very sparse with many zero elements and a special block-band structure. The L-B-J algorithm, originally suggested by Laffargue (1990) and

refined by Boucekkine (1995) and Julliard (1996) explicitly takes account of this special structure to solve the equations efficiently, using Gaussian pivoting on the blocks.

The method proceeds in two stages. In the first stage, the Jacobian matrix is transformed into an upper block-triangular structure, eliminating the blocks below the diagonal, first using the recursion

$$\text{subtract } \mathbf{B}_t^j \times \text{rows of block } t-j \text{ from rows of block } t$$

from $j = p^*, p^* - 1, \cdots, 1$, where $p^* = \min(p, t-1)$ and then replacing the block on the diagonal by the identity matrix by the operation

$$\text{premultiply rows of block } t \text{ by the inverse of the diagonal block } \mathbf{J}_t^*$$

where $\mathbf{J}_t^*$ is the diagonal block $\mathbf{J}_t$ after transformation by the recursive set of Gaussian eliminations. This step is applied, period by period, from $t = 1$ through to $t = T$. The first stage transforms the Jacobian matrix to an upper block-triangular structure. In the second stage, this structure is simply solved recursively, block by block, from period T down to period 1.

Not only does the algorithm exploit the block structure to minimise the calculations involved in solving the equations. It also economises on storage. The only blocks that need to be stored are those corresponding to the lead coefficients $\mathbf{F}_t^i$, $i = 1, \cdots, q$ so that storage is reduced from $nT \times nT$ to $nT \times nq$. This can be reduced further by dropping any columns in $\mathbf{F}_t^i$, corresponding to variables that never appear with a lead.

2.1 Terminal conditions in the L-B-J algorithm

One important issue that is neglected in the L-B-J algorithm is that of model *terminal conditions*. A solution of the model (1) requires that values are supplied for the variables $\mathbf{y}_{T+1}, \cdots, \mathbf{y}_{T+q}$ that lie outside of the solution period. One possibility is to set these terminal values to fixed exogenous values, in an analogous manner to the way that initial conditions are usually treated. In this case, the Newton algorithm is not affected since the derivatives will then be zero.

However, we may prefer to specify equations to define the terminal values. These may be equations defining equilibrium values for the variables, derived from a deterministic steady state solution of the model. (Note that choosing a deterministic steady state solution as a terminal condition imposes the

condition that the model has returned to equilibrium by period $T+1$, which may be unrealistic). Alternatively, when a model has no steady state (or one cannot easily be found), it is possible to specify a simple rule as a terminal condition such as a constant level $\mathbf{y}_{T+j} = \mathbf{y}_T$, $j = 1, \cdots, q$ or constant growth rate $\mathbf{y}_{T+j} = \mathbf{y}_T^{j+1}\mathbf{y}_{T-1}^{-j}$, $j = 1, \cdots, q$.

Imposing different terminal conditions may lead to different model solutions. The issues may be illustrated by considering the simple linear rational expectations model of Muth (1961) defined by the equation

$$p_t = \alpha p_{t+1}^e + \varepsilon_t$$

where p_{t+1}^e is the expected value of p_{t+1} formed in period t and ε_t is a stochastic driving process. Solving the model deterministically, we make the assumption of model consistent expectations,

$$p_{t+1}^e = p_{t+1}$$

and set the shochastic driving process to zero. Attempting to solve the model over a finite horizon, $t = 1, \cdots, T$, it is clear that the solution path is determined entirely by the terminal condition. In this case, imposing a constant level or constant growth rate terminal rule leads to a singularity and no solution is possible. The deterministic steady state of the model is given by $p = 0$ (except when $a = 1$ in which case there are an infinite number of steady states) and imposing this as the terminal condition gives $p_t = 0$ for all t. Setting any other constant terminal value, the solution values are determined by the equation

$$p_t = \frac{1}{a} p_{t+1}, \quad t = T-1, \cdots, 1.$$

Fair and Taylor (1983) recommended that, when solving a model, the influence of the terminal conditions be tested by extending the solution period until further extensions had no effect on the time path of the model variables over the original period of interest. Unfortunately, in practice these so-called type 3 iterations in the Fair-Taylor solution procedure are rarely conducted.

It is possible to generalise the L-B-J algorithm to incorporate terminal condition equations. In general, the model terminal conditions can be defined by the set of equations

$$\mathbf{h}_{T+j}(\mathbf{y}_{T+j}, \mathbf{y}_{T+j-1}, \cdots, \mathbf{y}_{T+j-r}, \mathbf{x}_{T+j}; \boldsymbol{\phi}) = \mathbf{0} \quad , \quad j = 1, \cdots, q \qquad (4)$$

where $\mathbf{h}_{T+j}$ is an $n \times 1$ vector valued function, $\boldsymbol{\phi}$ is a vector of parameters and r is the longest lag appearing in any of the terminal equations. (There can be no leads in the terminal equations of course as they must be backward-looking). Note that (4) allows a terminal condition to be defined for every endogenous variable in the model. In practice, terminal conditions need only be specified for variables that appear in the model with a lead.

The complete set of equations (1) and (4) may now be solved as a stacked-Newton system, using the L-B-J algorithm. The Newton equations (3) need to be replaced by the augmented set of equations

$$\mathbf{J}^{+}\begin{bmatrix} \mathbf{y}^{s}-\mathbf{y}^{s-1} \\ \mathbf{y}^{s}_{T+1}-\mathbf{y}^{s-1}_{T+1} \\ \vdots \\ \mathbf{y}^{s}_{T+q}-\mathbf{y}^{s-1}_{T+q} \end{bmatrix} = \begin{bmatrix} -\mathbf{f}(\mathbf{y}^{s-1}) \\ -\mathbf{h}_{T+1}(\mathbf{y}^{s-1}_{T+1}) \\ \vdots \\ -\mathbf{h}_{T+q}(\mathbf{y}^{s-1}_{T+q}) \end{bmatrix} \tag{5}$$

and the Jacobian matrix (2) replaced by the augmented matrix

$$\mathbf{J}^{+} = \begin{bmatrix}
\mathbf{J}_1 & \mathbf{F}^1_1 & \cdots & \mathbf{F}^q_1 & & & & & & & \\
\mathbf{B}^1_2 & \mathbf{J}_2 & \mathbf{F}^1_2 & \cdots & \mathbf{F}^q_2 & & & & & & \\
\vdots & \ddots & \ddots & \ddots & \cdots & \ddots & & & & & \\
\mathbf{B}^p_p & \cdots & \mathbf{B}^1_p & \mathbf{J}_p & \ddots & \cdots & \ddots & & & & \\
 & \ddots & \cdots & \ddots & \ddots & \ddots & \cdots & \mathbf{F}^q_{T-k} & & & \\
 & & \ddots & \cdots & \ddots & \ddots & \ddots & \vdots & & & \\
 & & & \ddots & \cdots & \ddots & \mathbf{J}_{T-1} & \mathbf{F}^1_{T-1} & & & \\
 & & & & \mathbf{B}^p_T & \cdots & \mathbf{B}^1_T & \mathbf{J}_T & & & \\
 & & & & & \mathbf{H}^r_{T+1} & \cdots & \mathbf{H}^1_{T+1} & \mathbf{H}^0_{T+1} & & \\
 & & & & & & \ddots & \cdots & \ddots & \ddots & \\
 & & & & & & & \mathbf{H}^r_{T+q} & \cdots & \mathbf{H}^1_{T+q} & \mathbf{H}^0_{T+q}
\end{bmatrix}$$

where

$$\mathbf{H}^i_{T+j} = \frac{\partial \mathbf{h}_{T+j}}{\partial \mathbf{y}'_{T+j-i}}, \quad i = 0, \cdots, r, \quad j = 1, \cdots, q.$$

The standard L-B-J recursions can be applied to this augmented system in the usual way.

Explicitly incorporating terminal conditions into the model in this way has several advantages. Firstly, suppose that we wish to impose deterministic steady state terminal conditions. In simulations in which the model

is subject to a permanent exogenous shock, the deterministic model steady state will change. Without explicitly specified terminal conditions, it would be necessary to run two simulations in this case: the first on a steady state version of the model to determine the new steady state values and the second on the dynamic version of the model, imposing the exogenous terminal values determined from the first simulation. With explicitly specified terminal conditions, the correct terminal values will automatically be calculated. A second advantage of explicit terminal conditions is that it makes it possible to solve models even when the 'correct' terminal value is not known, by using a simple rule to impose constant level or constant growth at the terminal date.

3 A stochastic growth model

This section describes a simple stochastic growth model, originally proposed by Christopher Sims, that was used by Taylor and Uhlig (1990) and other authors in the same journal issue to compare a number of different model solution methods. Although very simple, it does not have an analytic solution, except in a special case.

Agents are assumed to be infinitely lived and to maximise lifetime expected utility subject to a budget constraint. A constant relative risk aversion utility function is assumed

$$u(C_t) = (1-\tau)^{-1} C_t^{1-\tau}$$

where C_t is consumption and τ is the coefficient of relative risk aversion $0 < \tau < 1$. Then, formally, agents solve the following problem:

$$\max E_0 \sum_{t=0}^{\infty} \beta^t (1-\tau)^{-1} C_t^{1-\tau} \tag{6}$$

subject to the resource constraint

$$C_t + K_t = \theta_t K_{t-1}^{\alpha} \tag{7}$$

where K_t is the end of period capital stock, and θ_t is technology. $1-\mu$ is the rate of capital depreciation, $0 \leq \mu \leq 1$ and β is the rate of time discount,

$0 < \beta < 1$.[2] We also impose the side-conditions that $C_t > 0$ and $K_t > 0$, for all t. Technology θ_t is assumed to be stochastic, following the autoregressive process

$$\ln \theta_t = \rho \ln \theta_{t-1} + \varepsilon_t \tag{8}$$

where ε_t is a serially uncorrelated normally distributed random variable with zero mean and constant variance σ^2.

The first order Euler condition for capital in this model is given by

$$C_t^{-\tau} = E_t[\beta C_{t+1}^{-\tau}(\mu + \alpha\theta_{t+1}K_t^{\alpha-1})] \tag{9}$$

The solution to this model is a decision rule for consumption and one for capital stock given by $C_t = f(K_{t-1}, \theta_t)$ and $K_t = g(K_{t-1}, \theta_t)$ respectively. In general the exact forms of functions $f(\cdot)$ and $g(\cdot)$ are not known and solutions must be found by numerical solution of the equations (7), (8) and (9) over a finite time horizon $t = 1, \cdots, T$.

An analytic expression for the deterministic long-run steady state of the full model can be evaluated by setting $\varepsilon_t = 0$, $\theta_{t-1} = \theta_{t+1} = \theta_t$, $K_{t-1} = K_t$, $E_t C_{t+1} = C_t$, and solving equations (7), (8) and (9). The solution is given by:

$$\begin{aligned} \theta^* &= 1 \\ K^* &= \left(\frac{\alpha\beta}{1-\beta\mu}\right)^{1/(1-\alpha)} \\ C^* &= \left(\frac{\alpha\beta}{1-\beta\mu}\right)^{\alpha/(1-\alpha)} + (\mu - 1)\left(\frac{\alpha\beta}{1-\beta\mu}\right)^{1/(1-\alpha)}. \end{aligned} \tag{10}$$

4 Simulations with the stochastic growth model

Stochastic simulations were performed with the stochastic growth model under alternative terminal condition assumptions. All simulations were carried out using *WinSolve*, solving the model for 1999 periods and using 1000 replications. Normally distributed shocks to the logarithm of technology (with variance σ^2) were applied for the first 1990 periods only, allowing the model 9 periods in which to settle down before the terminal condition is imposed. The following values were used for the model parameters: $\rho = .95$, $\alpha = .33$,

[2] To be precise, the orginal model proposed by Sims assumed no capital depreciation so that $\mu = 1$. However, allowing some depreciation does not materially complicate the model.

$\beta = .95$, $\mu = 0.7$, $\tau = 1$, and $\sigma = .01$. These values correspond to one of the low variance cases reported in den Haan and Marcet (1990), one of the papers in the same journal issue as Taylor and Uhlig (1990) that discusses solution of this model (using a completely different method).

Since the model has only a single expectational variable, C_{t+1}, and a maximum lead of one, only one terminal equation is needed. Three alternative equations were employed:

$$C_{T+1} = (\frac{\alpha\beta}{1-\beta\mu})^{\alpha/(1-\alpha)} + (\mu - 1)(\frac{\alpha\beta}{1-\beta\mu})^{1/(1-\alpha)} \tag{11}$$

which is the deterministic steady state solution for C,

$$C_{T+1} = C_T \tag{12}$$

which is a constant level terminal condition and

$$C_{T+1} = \frac{C_T^2}{C_{T-1}} \tag{13}$$

which is a constant growth terminal condition. In the latter two cases, the equations did not need to be specified explicitly since *WinSolve* provides them automatically as possible choices of terminal condition rule.

The results of the experiment are summarised in Figure 1, which displays the mean from each of the three simulations over the last 19 periods, where the influence of the terminal condition is strongest. The first (blue) line is the deterministic steady state terminal condition case where the terminal value is forced to the deterministic steady state value of 0.696135. The second (green) line is the constant level terminal condition case and the third (red) line is the constant growth terminal condition case.

It can be seen that the three different terminal conditions visibly influence the last 15 or so periods of the simulation. However, the differences are quite small and only occur in the fifth decimal place.

5 Conclusions

A new algorithm has been proposed that extends the popular Newton-based L-B-J algorithm for solving forward-looking nonlinear models to allow the specification of explicit terminal conditions defined by equations. This algorithm is implemented in the software package *WinSolve*. The effect of

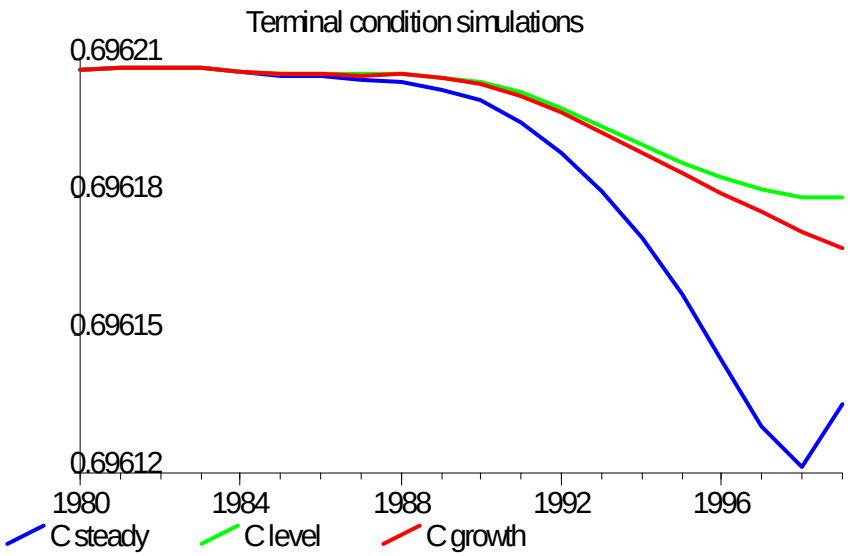


Figure 1: Simulation means for C_t with different terminal conditions

imposing different terminal conditions has been demonstrated in stochastic simulations with a simple stochastic growth model. For this model, the differences, while clearly visible, are rather small in scale although in other models, this may not always be the case.

References

[1] Boucekkine, R. (1995), 'An alternative methodology for solving nonlinear forward-looking models', *Journal of Economic Dynamics and Control*, 19, 711–734.

[2] den Hann, W.J. and A. Marcet (1990), 'Solving the stochastic growth model by parameterizing expectations', *Journal of Business and Economic Statistics*, 8, 31–34.

[3] Fair, R. C. and J. B. Taylor (1983), 'Solution and maximum likelihood estimation of dynamic nonlinear rational expectations models', *Econometrica*, 51, 1169–1186.

[4] Hollinger, P. (1996), 'The stacked-time simulator in TROLL: a robust algorithm for solving forward-looking models', mimeo, Intex Solutions, Needham, MA, USA.

[5] Juillard, M. (1996), 'DYNARE: a program for the resolution and simulation of dynamic models with forward variables through the use of a relaxation algorithm', CEPREMAP working paper No. 9602, Paris, France.

[6] Juillard, M., Laxton, D., McAdam, P. and Pioro, H. (1998), 'An algorithm competition: first-order iterations versus Newton-based techniques', *Journal of Economic Dynamics and Control*, 22, 1291–1318.

[7] Laffargue, J-P. (1990), 'Résolution d'un modèle macroéconomique avec anticipations rationnelles', *Annales d'Economie et de Statistique*, 17, 97–119.

[8] Muth, J.F. (1961), 'Rational expectations and the theory of price movements', *Econometrica*, 29, 315–335.

[9] Pierse, R.G. (2002), 'WinSolve: a users' guide', available at http://www.econ.surrey.ac.uk/winsolve/.

[10] Taylor, J.B. and H. Uhlig (1990), 'Solving nonlinear stochastic growth models: a comparison of alternative solution methods', *Journal of Business and Economic Statistics*, 8, 1–17.