Chapter 4

Small Analytical Models and WinSolve

4.1 Introduction

Many problems in economics are based on agents maximising an objective function subject to constraints. These typically lead to discounted dynamic programming problems for which Euler conditions can be derived but which in general are not amenable to analytic solution. To solve these models therefore requires numerical methods. In principle, any algorithm for solving nonlinear models involving both leads and lags of variables can be used to solve these models. In practice, however, the most commonly used solution algorithms such as *Fair-Taylor* have difficulty in solving them. In consequence, specialised algorithms have been proposed such as the *parameterised expectations* algorithm of den Haan and Marcet (1990). In addition, as these analytical models tend to be small, powerful solution methods such as the *Stacked Newton* method are feasible.

WinSolve now implements both a general *Stacked Newton* method and the den Haan and Marcet *parameterised expectations* algorithm. This makes it a powerful tool for solving small analytical models.

4.2 The Optimal Growth model

This section sets out a version of the optimal growth model originally developed by Ramsey (1928) and Koopmans (1965) *inter alia*. This model is developed in continuous time. A discrete time approximation is then made so that the model can be coded up in *WinSolve*.[1]

We assume that households are infinitely lived and that population grows at an exogenous rate n. Households solve the problem:

$$\max \int_{t=0}^{\infty} u[C(t)]e^{nt}e^{-\rho t}dt$$

where ρ is the rate of time preference with $\rho > n$, subject to a resource constraint. Assuming that production technology is labour augmenting and growing at an exogenous rate x, then this constraint can be written as

$$\dot{k} = f(k) - (n + x + \mu)k - c$$

where lower case denotes variables deflated by the effective labour supply, so that $f(k)$ is the production function in terms of capital per effective worker, and μ is the rate of depreciation of capital.

Assuming a utility function in the class of constant elasticity of intertemporal substitution (*CEIS*) utility functions

$$u[C(t)] = \frac{C^{1-\tau} - 1}{1 - \tau}$$

where $\tau = \sigma^{-1}$ and σ is the the elasticity of intertemporal substitution, then the first order condition leads to a consumption function

$$\frac{\dot{c}}{c} = \frac{1}{\tau}(f'(k) - \mu - \rho - \tau x) \ .$$

In steady state, $\dot{c} = \dot{k} = 0$ so that

$$f'(k^*) = \mu + \rho + \tau x$$

and

$$c^* = f(k^*) - (n + x + \mu)k^*$$

where c^* and k^* are the steady state values of c and k respectively.

The model can be discretised by the approximation

$$\dot{z} \cong \frac{z_{t+1} - z_t}{h}$$

where h is the discretisation factor. Assuming a Cobb-Douglas production function

$$f(k) = k^{\alpha}$$

[1] The *WinSolve* formulation of this model was originally developed by Michael Chui. I am grateful to him for providing me with the coded equations and for useful discussions.

and taking $h = 1$ leads to the discrete equations

$$k_{t+1} - k_t = k_t^\alpha - (n + x + \mu)k_t - c_t$$

and

$$\frac{c_{t+1} - c_t}{c_t} = \frac{1}{\tau}(\alpha k_t^{\alpha-1} - \mu - \rho - \tau x) \ .$$

This model can be coded in *WinSolve* as

```
k = k(−1) + k(−1) ^alpha − (n+x+mu) * k(−1) − c(−1) ;
c = c(1) / (1 + (alpha*k^(alpha−1)−mu−rho−tau*x)/tau) ;
```

Note that the first equation has been lagged while the second equation has been renormalised on c_t and so incorporates a lead. This normalisation reflects the saddlepath stability conditions in the model. The model can then be solved in *WinSolve* over any finite time horizon using the Stacked Newton solution method.

4.3 A Stochastic Growth Model

In this model agents are assumed to be infinitely lived and to maximise lifetime expected utility subject to a budget constraint. We assume a constant relative risk aversion (*CRRA*) utility function

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

where c_t is consumption and τ is the coefficient of relative risk aversion $0 < \tau < 1$.

Then formally agents solve the following problem:

$$\max E_0 \sum_{t=0}^{\infty} (1 - \tau)^{-1} c_t^{1-\tau} \beta^t \tag{4.1}$$

subject to the resource constraint

$$c_t + k_t = \theta_t k_{t-1} \alpha + \mu k_{t-1} \tag{4.2}$$

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, $0 < \beta < 1$.

Technology θ_t is assumed to be stochastic, following the process

$$\ln \theta_t = \rho \ln \theta_{t-1} + \varepsilon_t \tag{4.3}$$

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

The Euler equation for this model is given by

$$c_t^{-\tau} = \beta E_t[c_{t+1}^{-\tau}(\theta_{t+1}\alpha k_t^{\alpha-1} + \mu)] \tag{4.4}$$

The solution to this problem is a decision rule for consumption and for the capital stock given by $c_t = f(k_{t-1}, \theta_t)$ and $k_t = g(k_{t-1}, \theta_t)$ respectively. However, the exact form of $f(\cdot)$ and $g(\cdot)$ is not known analytically as there is no closed-form solution to this model. Solutions must be found by numerical solution of the equations (4.4), (4.2) and (4.3).

Taylor and Uhlig (1990) and the subsequent papers in the same issue of the Journal of Business and Economic Statistics compare different numerical techniques for solving this model.

4.3.1 A special case: Brock-Mirman model

In the special case where the utility function is logarithmic ($\tau = 1$) and there is full depreciation ($\mu = 0$), then the model simplifies to

$$\max E_0 \sum_{t=0}^{\infty} \ln c_t \beta^t$$

subject to

$$c_t + k_t = \theta_t k_{t-1} \alpha$$

and the Euler condition becomes

$$c_t^{-1} = \beta E_t[c_{t+1}^{-1}(\theta_{t+1}\alpha k_t^{\alpha-1})] \ .$$

This model is described in Brock and Mirman (1972) and known as the *Brock-Mirman* economy. In this case there is a simple closed-form solution (see for example Sargent (1987) p122) given by

$$k_t = \alpha\beta k_{t-1}^{a}\theta_t$$

and

$$c_t = (1 - \alpha\beta)k_{t-1}\alpha\theta_t \ .$$

4.4 Parameterised Expectations

The problem in solving the stochastic growth model is in finding the expectation

$$E_t[y_{t+1}] \tag{4.5}$$

in (4.4) where

$$y_t = c_t^{-\tau}(\theta_t \alpha k_{t-1}^{\alpha-1} + \mu)\,.$$

This expectation is a function of the state variables $x_t = \{k_{t-1},\, \theta_t\}$ but its form is unknown. Note that on the assumption of model consistent expectations,

$$E_t[y_{t+1}] = y_{t+1}\ .$$

den Haan and Marcet (1990) propose a general method for solving models by approximating expectations such as (4.5) using a functional form

$$\psi_t(\mathbf{x}_t; \boldsymbol{\delta})$$

where $\mathbf{x}_t$ is a $p \times 1$ vector of state variables and $\boldsymbol{\delta}$ is a $k \times 1$ vector of parameters. These parameters are chosen such as to minimise the sum of squared residuals

$$\min_{\boldsymbol{\delta}} \sum_{t=1}^{T} (y_{t+1} - \psi_t(\mathbf{x}_t; \boldsymbol{\delta}))^2\ .$$

This is simply a nonlinear least squares problem and can be solved using Newton's method by iterating on

$$\boldsymbol{\delta}^s = \boldsymbol{\delta}^{s-1} + (\boldsymbol{\Psi}'_{s-1}\boldsymbol{\Psi}_{s-1})^{-1}\boldsymbol{\Psi}'_{s-1}(y_{+1} - \boldsymbol{\psi}(\mathbf{x}; \boldsymbol{\delta}^{\mathbf{s-1}}))$$

where

$$\boldsymbol{\Psi}_{s-1} = \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\delta}'}$$

is the $T \times k$ matrix of derivatives of $\boldsymbol{\psi}$ with respect to the parameters $\boldsymbol{\delta}$ evaluated at iteration $s-1$.

The functional form of $\boldsymbol{\psi}$ should be chosen so as to be able to approximate the expectation as closely as possible. den Haan and Marcet suggest the class of *power functions*

$$\exp P_n(\ln(\mathbf{x}))$$

where P_n is a polynomial function of degree n. With large enough n, this class of functions can approximate any function $R_+^p \to R_+$ arbitrarily well. For the stochastic growth model they suggest

$$\psi_t(k_{t-1}, \theta_t; \boldsymbol{\delta}) = \delta_1 k_{t-1}^{\delta_2} \theta_t^{\delta_3} = \exp P_1(\ln k_{t-1}, \ln \theta_t)$$

but also consider higher order power functions.

den Haan and Marcet (1994) propose a test of solution accuracy that can be applied to the method of parameterised expectations. This is implemented by increasing the degree of the power function and testing the significance of the additional coefficients.

4.5 Parameterised expectations in WinSolve

WinSolve implements the den Haan and Marcet algorithm for parameterising expectations as a function within the model definition language. The stochastic growth model can be written as

*log(theta) = rho*log(theta(−1)) + norm(sigma2) ;*
*c = (beta * cexp(1)) ^ (1 / tau) ;*
*k = theta * k(−1) ^ alpha + mu * k(−1) − c ;*
*cexp = C ^ tau * (alpha * theta * k(−1) ^ (alpha − 1) + mu);*

The second equation corresponds to the Euler condition (4.4) where *cexp(1)* is the forward expectation of *cexp* which is defined by the fourth equation. This expectation can be parameterised by replacing the second equation by

*c = (beta * parexp(cexp(1) , k(−1) , theta, 1 , 2)) ^(1 / tau) ;*

The *WinSolve* function *parexp()* takes arguments defined by

$$parexp(y, x_1, \cdots, x_p\ [, \delta_1, \cdots, \delta_k]\ , n, p)$$

where y is the expectation to be parameterised, $x_1, \cdots, x_p$ are the state variables, n is the order of the power function and p is the number of state variables. $\delta_1, \cdots, \delta_k$ represent *optional* initial values for the parameters of the power function. Good initial values will improve the speed of convergence of the method.

Note that parameterising expectations does not require a separate solution algorithm in *WinSolve* and either the *Fair-Taylor* or *Stacked Newton* methods may be used. In particular, the model may include some expectations that are parameterised and others that are not. However, when all model expectations are parameterised, then apart from the function *parexp()*, the model is backward looking, so that the parameterised expectations algorithm will be doing all the work. Iteration will continue until a fixed point is found when the output from *parexp()* will stop changing. Convergence criteria can be set as for any other model.

Once a model has been solved with parameterised expectations, *WinSolve* will save the solution values of the parameter vector $\boldsymbol{\delta}$ and will use these as starting values in subsequent solutions. This will speed up convergence in these subsequent runs.

4.6 The Stacked Newton method

Consider the general nonlinear set of equations

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

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, $\mathbf{f}$ is an $n \times 1$ vector valued function and $\boldsymbol{\theta}$ is a vector of parameters. This 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 has a special structure and looks like

$$\mathbf{J}^* = \begin{bmatrix} \mathbf{J}_1 & \mathbf{F}_1^1 & \cdots & \mathbf{F}_1^k & & & & \\ \mathbf{B}_2^1 & \mathbf{J}_2 & \mathbf{F}_2^1 & \cdots & \ddots & & & \\ \vdots & \mathbf{B}_3^1 & \ddots & \ddots & \cdots & \ddots & & \\ \mathbf{B}_p^p & \cdots & \ddots & \ddots & \ddots & \cdots & \ddots & \\ & \ddots & \cdots & \ddots & \ddots & \ddots & \cdots & \mathbf{F}_{T-k}^k \\ & & \ddots & \cdots & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & \cdots & \ddots & \mathbf{J}_{T-1} & \mathbf{F}_{T-1}^1 \\ & & & & \mathbf{B}_T^p & \cdots & \mathbf{B}_T^1 & \mathbf{J}_T \end{bmatrix}$$

where

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

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

The *Stacked Newton method* applies Newton's method to the stacked system. This involves, in iteration s, solving the set of nT equations

$$\mathbf{J}^*(\mathbf{y}^s - \mathbf{y}^{s-1}) = -\mathbf{f}(\mathbf{y}^{s-1}) \tag{4.6}$$

where $\mathbf{y}$ is the $nT \times 1$ vector of stacked values of the endogenous variables. These equations can be solved efficiently, taking account of the special form of the Jacobian matrix $\mathbf{J}^*$ and minimising storage requirements, following the approach suggested by Laffargue (1990) and Boucekkine (1995).

One important issue is that of *terminal conditions*. Solution requires values of the variables $\mathbf{y}_{T+1}, \cdots, \mathbf{y}_{T+k}$ which are outside the solution period. When these are set to fixed exogenous values, then they are analogous to initial conditions and do not affect the Newton algorithm. However, terminal conditions are often set according to an equation, either an automatic rule such as constant level $\mathbf{y}_{T+j} = \mathbf{y}_T, \; j = 1, \cdots, k$ or constant growth rate $\mathbf{y}_{T+j} = \mathbf{y}_T^{j+1} \mathbf{y}_{T-1}^{-j}, \; j = 1, \cdots, k$, or a user-defined equilibrium condition.

In this case, the set of equations to be solved (4.6) needs to be supplemented by nk rows of the form

$$\overline{\mathbf{J}}_{T+j}(\mathbf{y}_{T+j}^s - \mathbf{y}_{T+j}^{s-1}) = -\overline{\mathbf{f}}(\mathbf{y}_{T+j}^{s-1}) \quad , \quad j = 1, \cdots, k$$

where

$$\overline{\mathbf{J}}_{T+j} = \begin{bmatrix} \mathbf{0}_{n \times (T+j-q-1)n} & \overline{\mathbf{B}}_{T+j}^q & \cdots & \overline{\mathbf{B}}_{T+j}^1 & \mathbf{I}_n & \mathbf{0}_{n \times (k-j)n} \end{bmatrix}$$

is of dimension $n \times n(T+k)$ with

$$\overline{\mathbf{B}}_{T+j}^i = \frac{\partial \overline{\mathbf{f}}}{\partial \mathbf{y}'_{T+j-i}} \; .$$

Note that the terminal condition equations must be predetermined so that only lagged variables can appear in $\overline{\mathbf{f}}$.

WinSolve implements the Stacked Newton algorithm using analytic derivatives evaluated automatically and taking into account any terminal conditions defined by equations.

4.7 Further reading

Pierse (1997) is a concise review of solution methods for nonlinear models. Taylor and Uhlig (and the following 10 papers in the same volume) is an interesting comparison of different ways of solving the stochastic growth model set out here. Obstfeld and Rogoff (1996) has a good discussion of the discrete time version of the Ramsay-Koopmans model.

Bibliography

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

[2] Brock, W.A. and L. Mirman (1972), 'Optimal economic growth and uncertainty: the discounted case', *Journal of Economic Theory*, 4, 479–513.

[3] 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.

[4] den Haan, W.J. and A. Marcet (1994), 'Accuracy in simulations', *Review of Economic Studies*, 61, 3–17.

[5] Koopmans, T.C. (1965), 'On the concept of optimal economic growth', in *The Econometric Approach to Develoment Planning*, North Holland, Amsterdam.

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

[7] Obstfeld, M. and K. Rogoff (1996) *Foundations of International Macroeconomics*, The MIT Press, Cambridge, MA.

[8] Pierse, R.G. (1997), 'Solution methods for nonlinear models', lecture notes, Department of Economics, University of Surrey.

[9] Ramsay, F.P. (1928), 'A mathematical theory of savings', *Economic Journal*, 38, 543–559.

[10] Sargent, T.J. (1987), *Dynamic Macroeconomic Theory*, Harvard University Press, Cambridge MA.

[11] 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.