

WinSolve Version 4: A tutorial illustrating some of the new features*

Richard G. Pierse

November 2007

1 Introduction

This tutorial illustrates a few of the new features available in *WinSolve Version 4* (Pierse 2007). The model used in the tutorial is a simple *dynamic stochastic general equilibrium (DSGE)* model, specifically the stochastic growth model considered in Taylor and Uhlig (1990) and the papers immediately following in the same issue of the *Journal of Business and Economic Statistics*, where alternative solution techniques for this model are discussed. There are two broad solution approaches to this class of models. Firstly they can be solved directly using non-linear deterministic methods, either first order methods such as Fair-Taylor (as discussed in Gagnon (1990)) or derivative methods such as the stacked-Newton method (as in Pierse (2002)). Alternatively, they can first be approximated by a simpler problem and then the simpler problem solved using an appropriate method. Many alternative approximation methods have been suggested. The most popular involves the linearisation (or log-linearisation) of the model, followed by the solution of the linearised problem by the method of Blanchard and Kahn (1980) or the more general methods of Klein (2000), Sims (2002) or King and Watson (1998) and (2002). Applications of a linearisation approach to the stochastic growth model are discussed in the papers of McGratten (1990) and Christiano (1990). Other approximation approaches include the parameterised expectations approach of den Haan and Marcet (1990) and (1994) that is implemented in *WinSolve Version 3*. This approach explicitly approximates decision rules by power functions of arbitrary order, the coefficients of

*Parts of this paper were prepared for presentations at City University, London in February 2005 and at a HKMA workshop in Hong Kong in April 2006. I am grateful for comments received from participants at these presentations.

which are estimated from the model. A related approach is the perturbation method. This also approximates decision rules but is based on the application of Taylor's theorem and the implicit function theorem. Perturbation methods have been advocated by Judd (1996, 1998) and used by Jin and Judd (2002), Collard and Julliard (2001), Schmitt-Grohé and Uribe (2004) and Kim and Kim (2003) among others.

Version 3 of *WinSolve* (Pierse (2001)) provided non-linear solution methods (Fair-Taylor and stacked-Newton) and one approximation method, the method of parameterised expectations. Version 4 of *WinSolve* (Pierse (2007)) now provides, in addition, linear and log-linear approximation methods. A key feature of the implementation of these methods in *WinSolve* is that the linear or log-linear approximation is done automatically by the software, and the model transformed into a generalised state space form. This eliminates the need for the user to derive the approximation by hand as is necessary in order to use the *matlab* code of Klein, Sims or King and Watson. The linearised form is then solved using the generalised real Schur decomposition (*qz algorithm*) and decision rules are computed. This method corresponds exactly with the first-order perturbation method. (A later release of *WinSolve* will also compute higher order perturbation approximations).

Another new feature of *WinSolve Version 4*, illustrated in this tutorial, is the ability to produce 'true' fan charts by stochastic simulation. Fan charts have been popularised by the Bank of England and the National Institute of Economic and Social Research among others, as a way of displaying the uncertainty around model forecasts in the form of quantiles from a distribution. However, the way that this is usually done is by imposing an ad hoc skewed distribution (e.g. a two-part normal) around a central point forecast that is deterministic. *WinSolve* by contrast, can estimate the quantiles of the true empirical distribution of the forecast variables through stochastic simulation of the model. The tutorial illustrates the construction of a fanchart for the time path of consumption from stochastic simulation of the non-linear equations of the stochastic growth model.

Section 2 of this paper briefly describes the model to be used and derives some analytic results. Then the tutorial follows in Section 3, illustrating the use of both non-linear and linear (and log-linear) solution methods and the production of fancharts.

2 The model

In this tutorial, a simple stochastic growth model will be used to illustrate some of the different methods of solution of *DGSE* models. This model is

of interest because: (a) although it is extremely simple, it does not admit to analytic solution and (b) it has been extensively discussed in the literature, in particular in a special issue of the *Journal of Business and Economic Statistics* (see Taylor and Uhlig (1990) and subsequent articles).

The problem to be solved is:

$$\max E_0 \sum_{t=0}^{\infty} \beta^t (1 - \tau)^{-1} c_t^{1-\tau} \quad (2.1)$$

subject to

$$c_t + k_t = \theta_t k_{t-1}^\alpha + \mu k_{t-1} \quad (2.2)$$

$$\ln \theta_t = \rho \ln \theta_{t-1} + \varepsilon_t, \quad (2.3)$$

where c_t is consumption, k_t is the end of period capital stock, and θ_t is technology. Technology θ_t is assumed to be stochastic, following an autoregressive process with coefficient ρ , $|\rho| < 1$, where the shock ε_t is a serially uncorrelated normally distributed random variable with zero mean and constant variance σ^2 . The other parameters of the model are: τ , the coefficient of relative risk aversion $0 < \tau < 1$, μ , (one minus) the rate of capital depreciation, $0 \leq \mu \leq 1$ and β , the rate of time discount, $0 < \beta < 1$.

The Lagrangian of the problem is

$$L = \max E_0 \sum_{t=0}^{\infty} \beta^t \left[(1 - \tau)^{-1} c_t^{1-\tau} + \lambda_{1t} (c_t + k_t - \theta_t k_{t-1}^\alpha - \mu k_{t-1}) + \lambda_{2t} (\ln \theta_t - \rho \ln \theta_{t-1} - \varepsilon_t) \right]. \quad (2.4)$$

and first order conditions for a maximum are

$$\frac{\partial L_t}{\partial c_t} = \beta^t (c_t^{-\tau} + \lambda_{1t}) = 0 \quad (2.5)$$

$$\frac{\partial (L_t + L_{t+1})}{\partial k_t} = \beta^t \lambda_{1t} - \beta^{t+1} E_t [\lambda_{1t+1} (\theta_{t+1} \alpha k_t^{\alpha-1} + \mu)] = 0 \quad (2.6)$$

$$\frac{\partial (L_t + L_{t+1})}{\partial \theta_t} = \beta^t (\lambda_{2t} \theta_t^{-1} - \lambda_{1t} k_{t-1}^\alpha) - \beta^{t+1} \lambda_{2t+1} \rho \theta_t^{-1} = 0 \quad (2.7)$$

$$\frac{\partial L_t}{\partial \lambda_{1t}} = c_t + k_t - \theta_t k_{t-1}^\alpha - \mu k_{t-1} = 0 \quad (2.8)$$

$$\frac{\partial L_t}{\partial \lambda_{2t}} = \ln \theta_t - \rho \ln \theta_{t-1} - \varepsilon_t = 0 \quad (2.9)$$

Substituting (2.5) into (2.6) and rearranging gives the Euler equation

$$c_t^{-\tau} = \beta E_t [c_{t+1}^{-\tau} (\theta_{t+1} \alpha k_t^{\alpha-1} + \mu)]. \quad (2.10)$$

Solving the model is achieved either by explicitly solving the optimisation problem (2.4) or by solving the first order conditions defined by equations (2.8), (2.9) and (2.10). (Note that the other first order condition (2.7) simply serves to determine λ_{2t} and so can be neglected).

The solution of the optimisation problem will be a pair of nonlinear decision rules

$$\begin{aligned} c_t &= h_1(k_{t-1}, \theta_t) \\ k_t &= h_2(k_{t-1}, \theta_t). \end{aligned}$$

The deterministic steady state of the model can be derived analytically and is given by

$$\begin{aligned} c^* &= \left(\frac{\alpha\beta}{1-\beta\mu}\right)^{\alpha/(1-\alpha)} + (\mu-1)\left(\frac{\alpha\beta}{1-\beta\mu}\right)^{1/(1-\alpha)} \\ k^* &= \left(\frac{\alpha\beta}{1-\beta\mu}\right)^{1/(1-\alpha)} \\ \theta^* &= 1. \end{aligned} \tag{2.11}$$

2.1 Linear and log-linear decision rules

Christiano (1990) discusses solving the stochastic growth model by linear and log-linear approximation. For this simple model, he is able to derive explicit formulae for the linear (2.12) and log-linear (2.13) approximations to the decision rule for k_t taken around the deterministic steady state values k_* and c_* :

$$\begin{aligned} b_0 &= (1-\lambda)k_*, \quad b_1 = \lambda, \quad b_2 = \frac{q\lambda}{1-\beta\rho\lambda} \\ k_t &= b_0 + b_1\theta_t + b_2k_{t-1} \end{aligned} \tag{2.12}$$

$$\begin{aligned} a_0 &= (1-\lambda)\log k_*, \quad a_1 = \frac{q}{k_*} \frac{\lambda}{1-\beta\rho\lambda}, \quad a_2 = \lambda \\ \log k_t &= a_0 + a_1\log \theta_t + a_2\log k_{t-1} \end{aligned} \tag{2.13}$$

where

$$\begin{aligned} q &= \beta \left[(1-\rho) \left(\frac{c_*}{k_*} + 1 - \mu \right) + \frac{\rho\beta}{\tau} (\beta^{-1} - \mu) \frac{c_*}{k_*} \right] k_* \\ \psi &= 1 + \beta^{-1} + \frac{(1-\alpha)(1-\beta\mu)}{\tau} \frac{c_*}{k_*} \end{aligned}$$

and

$$\lambda = \frac{\psi - \sqrt{\psi^2 - 4/\beta}}{2}.$$

2.2 Non-linear solution methods

Non-linear solution methods solve the first order conditions (2.8) - (2.10) directly over a finite time horizon $t = 1, \dots, T$. The assumption of perfect foresight is made to reduce the problem to a deterministic one.

First-order non-linear solution methods such as Fair-Taylor require that the equations are normalised, with a different endogenous variable on the left-hand side of each equation. One such normalisation is

$$c_t = [\beta c_{t+1}^{-\tau} (\theta_{t+1} \alpha k_t^{\alpha-1} + \mu)]^{-1/\tau} \quad (2.14)$$

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

but an alternative normalisation is

$$c_t = \bar{\theta}_t k_{t-1}^\alpha + \mu k_{t-1} - k_t$$

$$k_t = k_t + (\bar{\theta}_{t-1} k_{t-1}^\alpha + \mu k_{t-1} - k_t)^{-\tau} - \beta (\mu + \bar{\theta}_t \alpha k_t^{\alpha-1}) (\bar{\theta}_t k_t^\alpha + \mu k_t - k_{t+1})^{-\tau}. \quad (2.15)$$

where $\bar{\theta}_{t-1} = \theta_t$. Note that in the first normalisation, c_t is the jump variable while in the second, k_t is the jump variable.

One important consideration in the solution of finite horizon non-linear *RE* models, is the setting of terminal conditions for the jump variables. In the first normalisation, a value needs to be set for c_{T+1} . In the second normalisation, a terminal value is needed for k_{T+1} . Terminal conditions can help pin down a solution, even in cases where no steady state or multiple steady states may exist.

3 Solving the model in WinSolve

WinSolve Version 4 provides several algorithms for solving *DSGE* models such as the stochastic growth model. Direct non-linear solution methods include the stacked-Newton and Fair-Taylor algorithms or the expectations can be parameterised using the den Haan and Marcet algorithm. By stochastic simulation, the empirical distribution of the time paths of variables c_t and k_t can be derived and these can be displayed in the form of fancharts. Alternatively, *WinSolve* can compute an automatic linear or non-linear approximation of the model and then solve the approximated model using the *qz* algorithm. This tutorial demonstrates these different solution methods on the stochastic growth model described in the previous section. It is recommended that the reader follow this tutorial while running *WinSolve*. The necessary equation files, *rbcnl.txt* and *rbclin.txt* can be found in the *dat* sub-directory of the main *WinSolve* installation.

3.1 Non-linear methods

In the *WinSolve* model definition language the stochastic growth model can be written as

$$\begin{aligned} ltheta &= rho * ltheta(-1) + norm(sigma * sigma); \quad theta = exp(ltheta); \\ c &= (beta * c(1))^{(-tau)} * (theta(1) * alpha * k^{(alpha-1)} + mu))^{(-1/tau)}; \quad (3.1) \\ k &= theta * k(-1)^alpha + mu * k(-1) - c; \end{aligned}$$

where the model parameters ρ , α , β , μ , τ , and σ have been coded as *WinSolve* parameters *rho*, *alpha*, *beta*, *mu*, *tau* and *sigma* respectively. The equation on the second line corresponds to the Euler condition (2.10). The model parameters have been set to the values $\rho = .95$, $\alpha = .33$, $\beta = .95$, $\mu = 1.0$, $\tau = 0.5$, and $\sigma = 0.1$ corresponding to case 1 in Taylor and Uhlig (1990), which is a high variance case.

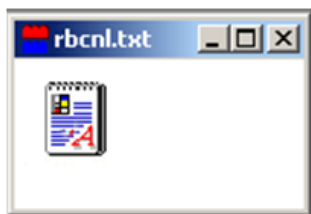



Figure 3.1: The model object

The model file to be opened is *rbcnl.txt*. The equations can be seen by double-clicking on the icon in the model object window (Figure 3.1) or selecting *Edit model* from the *File* menu, which opens the text editor.

3.1.1 Finding the deterministic steady state

The first step is to solve numerically for the *deterministic steady state* of the model. In order to do this, a new data file of 2500 undated observations needs to be created that will be replaced by the steady state solution. Click on the *Create new data file* icon , or select the *Create new data file...* option from the *Data* menu.

In the dialog box, select frequency *Undated* and data period 1 to 2500. Finally, check the *Initialise data* box to initialise all observations to zero. These option selections are illustrated in Figure 3.2. Click on the *OK* box to exit.

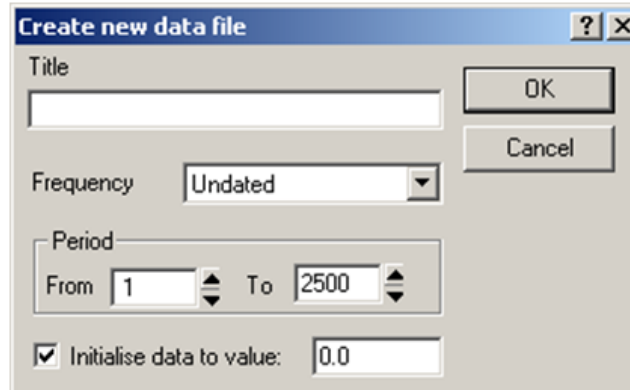




Figure 3.2: Create new data file dialog box

Before solving for the steady state, the variables c and k must be reset to a positive value, otherwise, the solution will fail. Go into the *Edit data / adjustments* dialog box and reset the data values for c and k to 1.

Now we can solve for the deterministic steady state of the model. Click on the *Solve model* icon  or select *Solve model ...* from the Solve menu. Choose *Steady state solution* from the Solution mode list as in Figure 3.3 and click *OK*. Viewing the results, it will be seen that the deterministic steady state values of k and c are 15.4864 and 2.46993 respectively. It can be verified that these values indeed correspond to the analytic solutions given in (2.11). Having found the deterministic steady state, this will now be used as the data base for further runs of the model. From the Solve menu, select the option *Set last solution as base*. A popup box will warn you that "*this will overwrite your current base values*". Click on *Yes* to agree to the change.

3.1.2 Dynamic solutions of the model

The model can now be solved dynamically, using the deterministic steady state as initial base values. Click on the *Solve model* icon  or select *Solve model ...* from the Solve menu. This time, choose *Dynamic model solution* from the Solution mode list. The non-linear model equations will be solved dynamically using the stacked Newton algorithm and the assumption of perfect foresight.

The model will now be solved again, this time using the parameterised expectations algorithm of den Haan and Marcet (1990). *WinSolve* implements the parameterised expectations algorithm through a function defined in the model definition language. For the case of the stochastic growth model, the

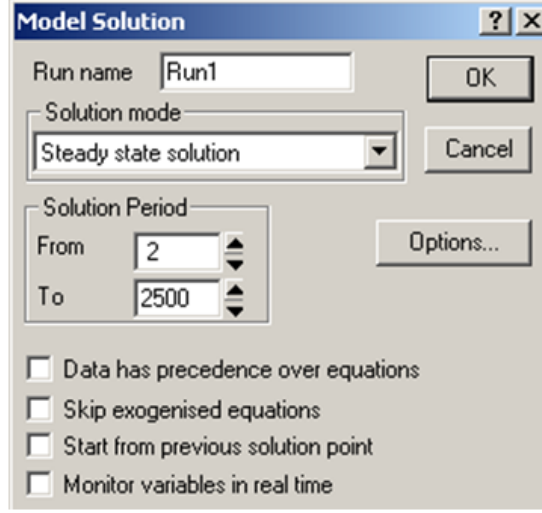


Figure 3.3: Model solution dialog box

expected value $c(1)$ can be parameterised by replacing the original equation for c in (3.1) with

$$\begin{aligned} cexp &= c^{\tau} * (\alpha * \theta * k(-1)^{(\alpha-1)+\mu}); \\ c &= (\beta * parexp(cexp(1), k(-1), \theta, 1, 2))^{(1/\tau)}; \end{aligned} \quad (3.2)$$

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

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

where y is the expectation to be parameterised, x_1, \dots, x_p are the state variables, n is the order of the power function and p is the number of state variables. $\delta_1, \dots, \delta_k$ represent *optional* initial values for the parameters of the power function. Good initial values will improve the speed of convergence of the method. Once a model has been solved with parameterised expectations, *WinSolve* will save the solution values of the parameter vector δ and will use these as starting values in subsequent solutions. This will speed up convergence in these subsequent runs.

Note that parameterising expectations does not require a separate solution algorithm in *WinSolve*. The *Fair-Taylor* method will be automatically selected but the parameterised expectations algorithm will be doing all the work since, apart from the function *parexp()*, the model is completely backward looking.

The equations (3.2) have already been included in the file *rbcnl.txt* as an alternative equation for c . To activate the parameterised expectations

algorithm, all we need to do is to switch to the alternative equation. Select the *Switch alternative equations* option in the *Assumptions* menu which brings up the dialog box in Figure (3.4). Choose the equation with description ‘1st

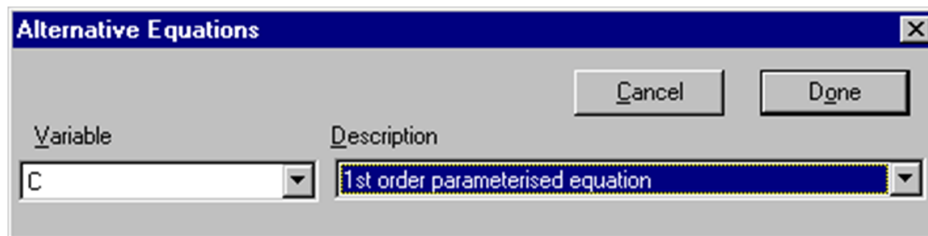



Figure 3.4: Switch alternative equations dialog box

order parameterised equation’ from the Description list box, and click *Done*.

Now the model must be solved again. *WinSolve* will automatically select an appropriate solution method for the parameterised expectation equation. Simply click on the Solve model icon  or select the *Solve model* option from the *Solve* menu, and then click on *OK* to commence model solution.

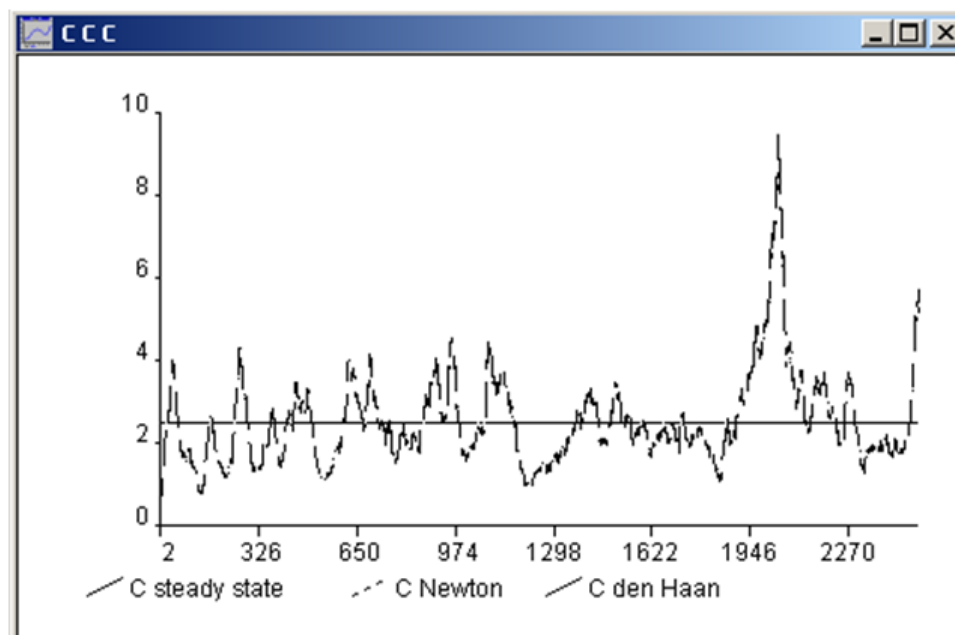


Figure 3.5: Nonlinear solutions of stochastic growth model

The three solutions to the model for variable c are graphed in Figure 3.5. It can be seen that there is very little difference between the stacked Newton

and parameterised expectations solutions. (Both were solved using the same drawing of shocks to technology).

3.1.3 Stochastic simulation

The two dynamic solutions considered so far have been based on a single drawing of the random shock to technology. Stochastic simulation solves the model repeatedly using different drawings of technology shocks. The average of these replications is then a consistent estimator of the expected value of the model variables and higher order moments such as variance, skew and kurtosis can also be computed. If desired, quantiles of the distribution can also be computed and these can be displayed graphically in the form of a fan chart. Note that this is a true fanchart of the empirical distribution of variables from a stochastic model. This is in contrast to the more familiar fancharts (as produced by the Bank of England for instance) in which an ad hoc skewed distribution is imposed around a central deterministic point forecast.

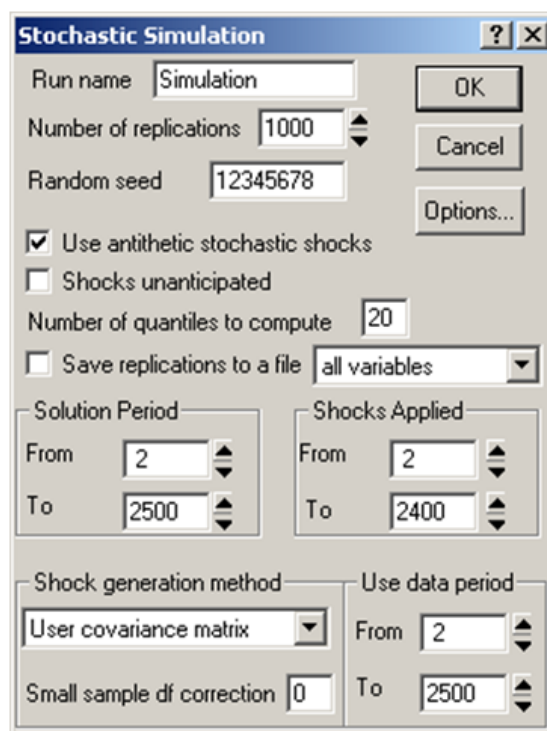



Figure 3.6: Stochastic simulation dialog box

To start stochastic simulation, click on the *Stochastic simulation* icon

 or select the *Stochastic simulation* option from the *Solve* menu. This opens the stochastic simulation dialog box (Figure 3.6). This dialog box is quite complicated and is worthy of some detailed explanation. The key selection is the *shock generation method*, which here should be specified as *User covariance matrix*. This option allows you to specify variances and covariances for the shocks to be applied to each equation. This is the only sensible choice here since only one equation, that for *ltheta*, is stochastic with known variance is $\sigma^2 = 0.01$. Other shock generation methods, such as *Cholesky* and *bootstrap* are available for empirically estimated models.

It is possible to restrict the period over which stochastic shocks are applied. This is particularly useful for models such as this with rational expectations where the terminal condition is a long run steady state condition since, if shocks are applied right up to the terminal date, it is difficult for the model to satisfy the terminal condition and convergence may be a problem. In this case it is sensible to restrict the period to apply shocks to be 2 (the first solution period) to 2400.

The number of replications has been set to 1000. This is a reasonable number of replications for this simulation, although greater precision could be achieved by increasing the replications at the cost of greater execution time. The option to *use antithetic variables* box has been checked. This forces the distribution of generated shocks to be (exactly) symmetric. However, in a non-linear model, it does not guarantee that the distribution of the model variables will be symmetric.

The random number seed initialises the random number generator used for generating pseudo-random shocks. Two stochastic simulations, over the same period and using the same random number seed, will generate the same results. This is useful when a simulation needs to be repeated at a later date. Changing the seed will result in a new set of random drawings. It is useful to make a note of the random number seed used in important simulations.

The stochastic simulation box has an option to specify that simulation quantiles are to be computed. This allows a complete probability distribution to be estimated (and graphed in a fan chart). This option is unset by default because it can be expensive since it necessitates that all replications are written to a temporary file and then read back in order to compute the necessary quantiles. The input box specifies the number of quantiles to be computed. The value of 20 allows quantiles in 5% intervals from 5% to 95%. The *save replications to a file* option may be used in conjunction with the quantiles option to save replications permanently for use outside *WinSolve*. Note that if only a single variable is required, it is considerably cheaper to specify the variable name than accept the default of saving all variables. Once

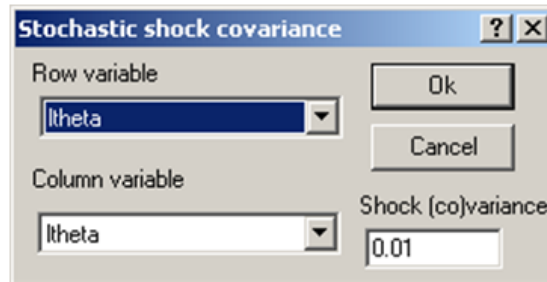


Figure 3.7: Stochastic shock covariance dialog box

all the desired options have been set in the dialog box, click the *OK* button to exit. Another dialog box will open (Figure 3.7) in which values should be entered for the non-zero elements of the shock covariance matrix. Select the variable *ltheta* from both row and column lists and enter the value 0.01. Then click the *OK* button, which opens the status box and starts the simulation. *WinSolve* will allow a maximum number of failed replications before giving up on a stochastic simulation. By default this number is set to 10 but it can be increased if necessary using the *maximum errors* control in the *solution options* dialog box. (This dialog is accessible from the stochastic simulation dialog box through the *Options* button.)

When the stochastic simulation has finished, the results are available to view in the usual way through the *New table/graph* box on the *Results* menu. After a stochastic simulation, various additional statistics are available to view: the simulation mean, standard deviation, skewness, mode and the $\pm 5\%$ simulation bands. In addition to these, if quantiles were requested, these are also available (the median of course is the 50% quantile) as well as two fan chart options. The *quantile fanchart* is a standard fanchart of quantiles, centred on the median of the distribution. The *modal fanchart* is an alternative fanchart, centred on the estimated mode of the distribution (as in the (ad hoc) fancharts reported by the Bank of England). Figure ?? presents a quantile fanchart for consumption. The bands represent 5% quantile intervals with the darkest central band showing the 45% to 55% interval including the median and the outermost bands the 5% and 95% bands. Note that the distribution is considerably skewed and that the median of the distribution (the centre of the darkest central band) is somewhat higher than the deterministic steady state for consumption in this model.

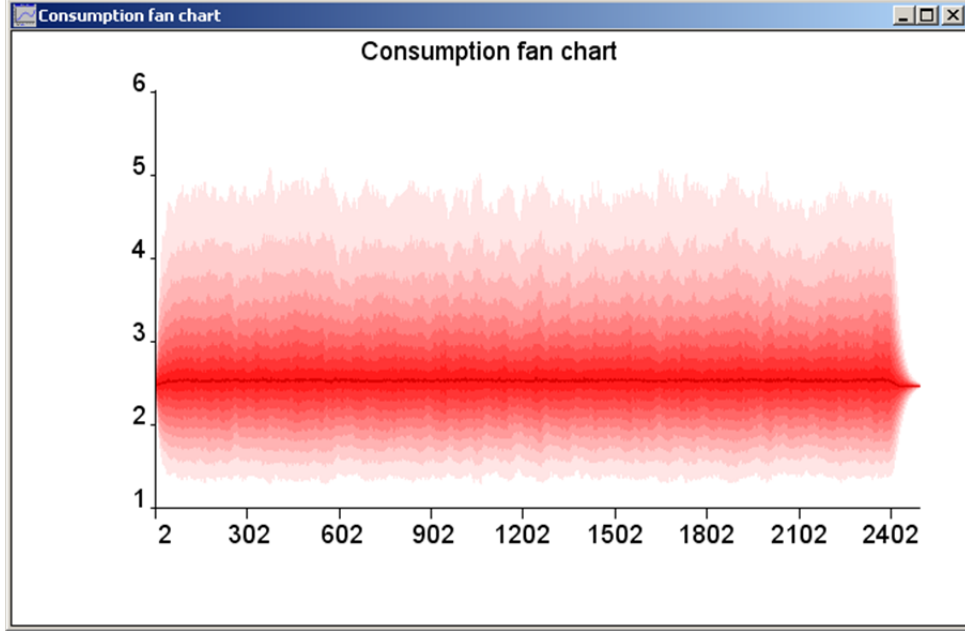


Figure 3.8: Consumption fan chart

3.2 Approximation methods

An alternative approach to solving the stochastic growth model is to find a linear or log-linear approximation of the non-linear model equations (3.1) and then use linear solution methods to solve this approximation to the model. This alternative solution method is a new feature of *WinSolve Version 4*. The approximation is done automatically by the program and can be computed at any desired data value for the model variables (usually the deterministic steady state). Then the resulting linear problem is solved using a *generalised Schur decomposition* (the *qz algorithm*) and the resulting decision rules displayed. It will be demonstrated that the decision rules computed automatically by *WinSolve* replicate exactly the decision rules computed analytically by Christiano (1990) for this model.

In order to proceed, we rewrite the model in a slightly different way, based on equation (2.15), where the variable c has been substituted out so that the only variables are k and θ . The rewritten model is given by

$$\begin{aligned} \log(\theta) &= \rho * \log(\theta(-1)) + \text{norm}(\sigma * \sigma); \\ k &= k + (\theta(-1) * k(-1)^\alpha + \mu * k(-1) - k)^\tau; \\ -\beta * (\mu + \theta * \alpha * k^{\alpha-1}) * (\theta * k^\alpha + \mu * k - k(1))^\tau; \end{aligned}$$

and these model equations are defined in file *rbclin.txt*.

As before, the first step is to solve for the deterministic steady state. Open the model and create a new data file of 2500 undated observations, initialised this time to 1. Then solve for the steady state and set this solution as the data base. It should be verified that the steady state solution for k is 15.4864 as before.

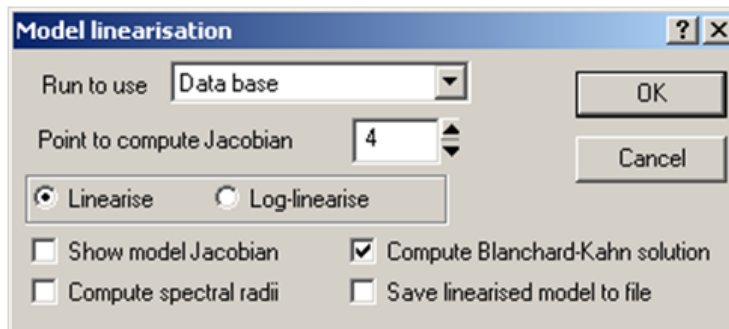


Figure 3.9: Model linearisation dialog box

To compute a linearisation around the deterministic steady state, select *Linear solution options...* from the *Solve* menu. The *Model linearisation dialog box* shown in Figure 3.9 will open. Choose *Data base* as the run to use and choose observation 4 (3 or greater) as the point at which to linearise. Select *linearise* or *log-linearise* as appropriate and uncheck all boxes except *Compute Blanchard-Kahn solution*. Then click *OK* to finish.

The output is presented in a table. Firstly, the (generalised) eigenvalues of the state space representation are shown and the Blanchard-Kahn conditions checked. Then the decision matrices are displayed. For the linear case, the eigenvalues are

$$1.14, \quad 0.95, \quad 0.92$$

so that there is one unstable eigenvalue and two stable eigenvalues and the conditions for a unique solution are satisfied. In this case there are no infinite eigenvalues so that the state space representation is non-singular. The slope coefficients of the decision rule are given in the table:

$$\begin{array}{cc} & \theta_t & k_{t-1} \\ k_t & 1.88723 & 0.923547 \end{array}$$

These slope coefficients are identical with those computed from the analytical formulae derived by Christiano in (2.12) which gives the rule as

$$k_t = 1.18398 + 1.88723\theta_t + 0.923547k_{t-1}.$$

Similarly, for the log-linear approximation, the eigenvalues are

$$1.14, \quad 0.95, \quad 0.92$$

as before and the coefficients of the decision rule are

$$\begin{array}{cc} \theta_t & k_{t-1} \\ k_t & 0.121863 \quad 0.923547 \end{array}$$

The Christiano decision rule in this case is

$$\log k_t = 0.209478 + 0.121863 \log \theta_t + 0.923547 \log k_{t-1}$$

so that, as before, the slope coefficients produced by *WinSolve* are identical with those derived by Christiano. Note that it is only possible to derive decision rules analytically in very simple cases, whereas the numerical procedure used by *WinSolve* can be applied to any model, however large.

References

- [1] Blanchard, O. and C. Kahn (1980), ‘The solution of linear difference models under rational expectations’, *Econometrica*, 48, 1305–1311.
- [2] Christiano, L.J. (1990), ‘Solving the stochastic growth model by linear-quadratic approximation and by value-function iteration’, *Journal of Business and Economic Statistics*, 8, 23–26.
- [3] Collard, F. and M. Juillard (2001), ‘Accuracy of stochastic perturbation methods: the case of asset pricing models’, *Journal of Economic Dynamics and Control*, 25, 979–999.
- [4] 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.
- [5] den Haan, W.J. and A. Marcet (1994), ‘Accuracy in simulations’, *Review of Economic Studies*, 61, 3–17.
- [6] Gagnon, J.E. (1990), ‘Solving the stochastic growth model by deterministic extended path’, *Journal of Business and Economic Statistics*, 8, 35–36.
- [7] Jin, H and K.L. Judd (2002), ‘Perturbation methods for general dynamic stochastic models’, mimeo.

- [8] Judd, K.L. (1996), ‘Approximation, perturbation and projection methods in economic analysis’, chapter 12 in H.M. Amman, D.A. Kendrick and J. Rust (eds.), *Handbook of Computational Economics, Volume I*, Elsevier Science, Amsterdam, 509–585.
- [9] Judd, K.L. (1998), *Numerical Methods in Economics*, The MIT Press, Cambridge, MA, USA.
- [10] Kim, J. and S.H. Kim (2003), ‘Spurious welfare reversals in international business cycle models’, *Journal of International Economics*, 60, 471–500.
- [11] King, R. G. and M. W. Watson (1998), ‘The solution of singular linear difference systems under rational expectations’, *International Economic Review*, 39, 1015–1026.
- [12] King, R.G. and M.W. Watson (2002), ‘System reduction and solution algorithms for singular linear difference systems under rational expectations’, *Computational Economics*, 20, 57–86.
- [13] Klein, P. (2000), ‘Using the generalized Schur form to solve a multivariate linear rational expectations model’, *Journal of Economic Dynamics and Control*, 24, 1405–1423.
- [14] McGratten, E.R. (1990), ‘Solving the stochastic growth model by linear-quadratic approximation’, *Journal of Business and Economic Statistics*, 8, 41–43.
- [15] Pierse, R. G. (2001), ‘WinSolve Version 3: An Introductory Guide’, Department of Economics, University of Surrey.
- [16] Pierse, R. G. (2002), ‘Solving a stochastic growth model using the Stacked-Newton method’, Department of Economics, University of Surrey.
- [17] Pierse, R. G. (2007), ‘WinSolve Version 4: An Introductory Guide’, Department of Economics, University of Surrey.
- [18] Schmitt-Grohé, S. and M. Uribe (2004), ‘Solving dynamic general equilibrium models using a second-order approximation to the policy function’, *Journal of Economic Dynamics and Control*, 28, 755–775.
- [19] Sims, C.A. (2002), ‘Solving linear rational expectations models’, *Computational Economics*, 20, 1–20.

- [20] 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.