An Evaluation of Alternative Approaches to Solving Large-Scale Macroeconomic Models

Ric D. Herbert[a] , Peter J. Stemp[b] and William E. Griffiths[c]

[a]School of Design Communication and Information Technology, The University of Newcastle, University Drive, Callaghan, NSW, 2308, Australia**;** email: ric.herbert@newcastle.edu.au

[b,c]Department of Economics, The University of Melbourne, Melbourne, Victoria 3010, Australia.

Abstract: This paper is concerned with assessing the time taken by the well known reverse-shooting and forward-shooting algorithms to solve large-scale macroeconomic models, which are characterized by having the particular property known as saddle-path instability. Given an arbitrary large-scale model about which we have limited information, how quick are the algorithms likely to be in solving this model? How are specific model properties likely to influence the time taken by a particular algorithm? We address this question using a range of investment models, which have been extended to allow for multi-dimensional specifications of the capital stock. Each algorithm presents a complicated exercise with a potentially unstable ordinary differential equation to be solved over a wide parameter space and involving a difficult search. There are a number of places where computational errors can be introduced and these errors can soon "blow-up". It is a good exercise on which to compare the two shooting methods. Our results provide insights into how the complexity of the solutions to a broad range of macroeconomic models increases with the dimensionality of the models. We also investigate additional questions that might be addressed in order to assess which approach is the best general way to find the stable trajectory of a model with saddle-path properties. We describe how econometric techniques could be used to summarize the likely success of competing algorithms when confronted with models exhibiting a range of properties.

Keywords: Macroeconomics; Saddlepath instability; Computational techniques; Investment models.

1. INTRODUCTION

Large-scale macroeconomic models play an important role in today's society. They are used for forecasting future values of a multitude of economic variables and for evaluating the effects of changes or proposed changes in government policy.

When a researcher is faced with the problem of choosing a particular algorithm for solving a macroeconomic model, it is not clear which algorithm is the most suitable. Sometimes algorithms fail; sometimes they are successful. Sometimes some algorithms are successful when others fail. The relative success of different algorithms depends on characteristics of the model such as dimensionality, degree of non-linearity and interactions between various models within the full model.

This paper is concerned with assessing the time taken by the well known reverse-shooting and forward-shooting algorithms to solve large-scale macroeconomic models, which are characterized by having the particular property known as saddle-path instability. We address this question using a range of investment models, which have been extended to allow for multi-dimensional specifications of the capital stock.

2. SOLVING A MODEL WITH SADDLE-PATH INSTABILITY

2.1. The Two Dimensional Problem

Consider the following two dimensional model, which has been linearized about $\mathbf{x}^*$:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\mathbf{x}(t) - \mathbf{x}^*) \tag{1}$$

where, throughout this paper, an asterisk denotes a steady state value and

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \text{ and } \mathbf{x}^* = \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} \tag{2}$$

Assume that **A** has one stable eigenvalue ($\lambda_1 < 0$) and one unstable eigenvalue ($\lambda_2 > 0$). Let $\underset{2x1}{\mathbf{v}(\lambda_i)}$ be the eigenvector of **A** associated with λ_i so that:

$$\mathbf{v}(\lambda_1) = \begin{bmatrix} a_{12} \\ \lambda_1 - a_{11} \end{bmatrix} \text{and } \mathbf{v}(\lambda_2) = \begin{bmatrix} a_{12} \\ \lambda_2 - a_{11} \end{bmatrix} \tag{3}$$

Then the solution to Equation (1) is given by:

$$\mathbf{x}(t) = \begin{bmatrix} a_{12} & a_{12} \\ \lambda_1 - a_{11} & \lambda_2 - a_{11} \end{bmatrix} \begin{bmatrix} C_1 \exp(\lambda_1 t) \\ C_2 \exp(\lambda_2 t) \end{bmatrix} \tag{4}$$

This solution has the property of saddle-path instability. A phase diagram for a typical

nonlinear model with saddle-path instability is given in Figure 1 below.

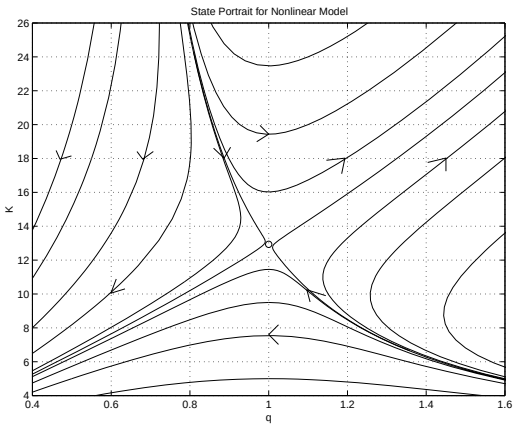

Figure 1. Phase Diagram for Two Dimensional Problem

In order to solve the model, it is necessary to derive the stable arm of the saddlepath. In the case of a nonlinear model this must be achieved by calibrating the model and using an appropriate algorithm to find the stable arm. The algorithms described in this paper are the well known reverse-shooting and forward-shooting algorithms.

2.2. The Higher Dimensional Problem

Next, consider the following model, also linearized about $\mathbf{x}^*$:

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x}(t) - \mathbf{x}^*) \tag{5}$$

where $\underset{mxm}{\mathbf{A}}$ is a square matrix, and $\underset{mx1}{\mathbf{x}(t)}$, $\underset{mx1}{\mathbf{x}^*}$ are column matrices.

Assume that $\mathbf{A}$ has s stable eigenvalues $(\lambda_1, \lambda_2, ..., \lambda_s)$ and u unstable eigenvalues $(\lambda_{s+1}, \lambda_{s+2}, ..., \lambda_{s+u})$ where $s+u=m$. Let $\underset{mx1}{\mathbf{v}(\lambda_i)}$ be the eigenvector of $\mathbf{A}$ associated with λ_i. Then, using the Jordan decomposition, it is possible to write the solution to this model in the form:

$$\mathbf{x}(t) - \mathbf{x}^* = \begin{bmatrix} \mathbf{v}(\lambda_1) & \mathbf{v}(\lambda_2) & \cdots & \mathbf{v}(\lambda_s) & \mathbf{v}(\lambda_{s+1}) & \mathbf{v}(\lambda_{s+2}) & \cdots & \mathbf{v}(\lambda_{s+u}) \end{bmatrix} \begin{bmatrix} C_1 \exp(\lambda_1 t) \\ C_2 \exp(\lambda_2 t) \\ \vdots \\ C_s \exp(\lambda_s t) \\ C_{s+1} \exp(\lambda_{s+1} t) \\ C_{s+2} \exp(\lambda_{s+2} t) \\ \vdots \\ C_{s+u} \exp(\lambda_{s+u} t) \end{bmatrix} \tag{6}$$

Blanchard and Kahn (1980) have shown that a stable solution for this model can be found as long as there are precisely as many "jump" variables as there are unstable eigenvalues. The initial values of these jump variables are determined by choosing the constants associated with the unstable eigenvalues equal to zero so that $C_{s+i} = 0$ for $i = 1, 2, ..., u$. Conversely, precisely m_s variables are predetermined by history. We refer to these latter variables as the "non-jump" variables. Therefore, history determines values for each C_i, where $i = 1, 2, ..., s$.

Solving the higher dimensional model is then equivalent to finding appropriate jumps to a stable path that converges to the new steady-state. This must be achieved by calibrating the model and using an appropriate algorithm to find the stable path.

2.3. Forward-Shooting and Reverse-Shooting

The forward-shooting solution to this model is given by first fixing an initial value for t, given by t_0, then searching over the values for $(C_1, C_2, ..., C_m)$ until a solution is found that is consistent with the initial values of the non-jump variables and arrives within a suitably small neighborhood of the steady-state, $\mathbf{x}^*$. In this sense, the forward-shooting solution is equivalent to searching over a space of dimension m.

To find the reverse-shooting solution, it is first necessary to write the model in reverse time, so that $\mathbf{z}(t) = \mathbf{x}(-t)$ and:

$$\mathbf{z}(t) - \mathbf{x}^* = \begin{bmatrix} \mathbf{v}(\lambda_1) & \mathbf{v}(\lambda_2) & \cdots & \mathbf{v}(\lambda_s) & \mathbf{v}(\lambda_{s+1}) & \mathbf{v}(\lambda_{s+2}) & \cdots & \mathbf{v}(\lambda_{s+u}) \end{bmatrix} \begin{bmatrix} C_1 \exp(-\lambda_1 t) \\ C_2 \exp(-\lambda_2 t) \\ \vdots \\ C_s \exp(-\lambda_s t) \\ C_{s+1} \exp(-\lambda_{s+1} t) \\ C_{s+2} \exp(-\lambda_{s+2} t) \\ \vdots \\ C_{s+u} \exp(-\lambda_{s+u} t) \end{bmatrix} \tag{7}$$

Without loss of generality, start at $t_0 = -N$ where N is a large positive number and choose $\mathbf{z}(t_0)$ close to $\mathbf{z}^*$. Then $C_i \exp(\lambda_i N)$ is close to zero for $i = 1, 2, ..., m$. If λ_i is an unstable eigenvalue, then $\exp(\lambda_i N)$ is a large positive number; hence C_i must be close to zero. On the other hand, if λ_i is a stable eigenvalue, then

$\exp(\lambda_i N)$ is close to zero; hence C_i can take any value.

Then, equation (7) reduces to:

$$\mathbf{z}(t)-\mathbf{x}^* = \mathbf{x}(-t)-\mathbf{x}^* = [\mathbf{v}(\lambda_1) \quad \mathbf{v}(\lambda_2) \quad \cdots \quad \mathbf{v}(\lambda_s)] \begin{bmatrix} C_1 \exp(-\lambda_1 t) \\ C_2 \exp(-\lambda_2 t) \\ \vdots \\ C_s \exp(-\lambda_s t) \end{bmatrix} \tag{8}$$

Then a solution is found by searching over the values for $(C_1, C_2, ..., C_s)$ until a solution is found that arrives within a suitably small neighborhood of the history-determined values for the non-jump variables of $\mathbf{x}(t)$ and hence for the non-jump variables of $\mathbf{z}(t)$. In this sense, the reverse-shooting solution is equivalent to searching over a space of dimension s.

Thus the reverse-shooting algorithm is more efficient than the forward-shooting algorithm in the sense that the reverse-shooting algorithm requires searching over a smaller dimensional space (since $s < s+u = m$).

In the next Section, we consider a special example of dimension 2n where $s = u = n$.

3. A SPECIFIC EXAMPLE

3.1. A Model of the Investment Decision

Consider the investment decision of a profit-maximizing firm with n types of capital along the lines of Hayashi (1982). The firm faces a Cobb-Douglas production technology. Also, adjustment costs are associated with the installation of new capital. The magnitude of these adjustment costs is governed by the magnitude of parameters, b_i. The decision of the firm can then be summarized as follows:

Choose the I_i's to maximize:

$$V = \int_0^\infty e^{-rt}[F(K_1, K_2, ..., K_n) - \sum_{i=1}^n I_i]dt \tag{9}$$

subject to

$$\dot{K}_i = I_i - b_i\left(\frac{I_i^2}{K_i}\right), \quad \text{for} \quad i = 1, 2, ..., n \tag{10}$$

$$K_i(0) = K_{io}, \quad \text{for} \quad i = 1, 2, ..., n \tag{11}$$

$$F(K_1, K_2, ..., K_n) = a\prod_{i=1}^n (K_i)^{\alpha_i} \tag{12}$$

where $\sum_{i=1}^n \alpha_i < 1$ and

K_i = real stock of capital of type i;

I_i = real level of investment of type i;

$F(K_1, K_2, ..., K_n)$ = real output;

r = real interest rate (assumed exogenous); and

a, b_i, α_i are exogenous parameters.

The dynamics of capital accumulation in the model reduce to the following set of equations:

$$\dot{q}_i = [r - b_i\left(\Lambda(b_i, q_i)\right)^2]q_i - F_i, \quad \text{for} \quad i = 1, 2, ..., n \tag{13}$$

$$\dot{K}_i = \Lambda(b_i, q_i)[1 - b_i\Lambda(b_i, q_i)]K_i, \quad \text{for} \quad i = 1, 2, ..., n \tag{14}$$

where

$$F_i = F_{K_i} = \frac{a\alpha_i}{K_i}\prod_{i=1}^n (K_i)^{\alpha_i} \tag{15}$$

$$\Lambda(b_i, q_i) = \frac{q_i - 1}{2b_i q_i} \tag{16}$$

The variables q_i are the co-state variables derived from the firm's optimization problem. These co-state variables are frequently referred to as Tobin's q.

The steady-state solutions of the model then reduce to the following:

$$q_i^* = 1, \quad \text{for} \quad i = 1, 2, ..., n \tag{17}$$

$$K_i^* = \left[\frac{r}{\alpha_i a}\prod_{j \neq i}\left(\frac{\alpha_i}{\alpha_j}\right)^{\alpha_j}\right]^{\frac{1}{\sum_i \alpha_i - 1}}, \quad \text{for} \quad i = 1, 2, ..., n \tag{18}$$

3.2. The Computational Problem

The computational problem we examine is the solution of the investment model above from a known meaningful steady state, $\mathbf{x}_0^*$, to a new known meaningful steady state, $\mathbf{x}^*$, after an exogenous shock in interest rates from r_0 to r. The problem is to find the unique trajectory (in q's and K's) from the initial steady state to the final steady state resulting from the shock.

The fundamental problem is to find the stable solutions for the following dynamical system:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}) \qquad (19)$$

where the state vector is given by:

$$\mathbf{x} = \left[\mathbf{q}, \mathbf{K}\right]^T = \left[q_1, q_2, ..., q_n, K_1, K_2, ..., K_n\right]^T \qquad (20)$$

and the parameter vector is given by:

$$\mathbf{p} = \left[a, \alpha_1, \alpha_2, ..., \alpha_n, b_1, b_2, ..., b_n\right]^T \qquad (21)$$

The dimensionality of the model is m (=2n). Notice that the model is autonomous so that calendar time plays no part in the solution.

The shock in interest rates determines the boundary conditions for the model and gives rise to the specific exercise we solve. Before the shock the model is at $\mathbf{x}_\mathbf{0}^* = \left[\mathbf{q}_\mathbf{0}^*, \mathbf{K}_\mathbf{0}^*\right]^T$ and evolves along a unique stable solution trajectory to $\mathbf{x}^* = \left[\mathbf{q}^*, \mathbf{K}^*\right]^T$ as given in equations (17-18). The problem is to find this stable trajectory. This trajectory must lie on the stable manifold, so that the vector of co-states, $\mathbf{q}$, must instantaneously jump onto the stable trajectory. Hence the initial conditions for the co-states are not known. The basic problem of this computational exercise is to find these co-state initial conditions.

The exercise is a two-point boundary value problem where the aim is to find the trajectory of the model. The exercise is difficult due to the unstable nature of the model problem. Basically for the reverse-shooting algorithm we need to search around points "near enough" to the final steady state so that the solution to the model has transient dynamics in reverse time that are forced onto the stable manifold and that also satisfy the appropriate initial conditions. For the forward-shooting algorithm we search around points at the initial values of the capital stocks, to find an initial value for the $\mathbf{q}$ vector so that the solution to the model has transient dynamics in forward time that pass "near enough" to the final steady-state. Both searches will determine a solution trajectory and initial conditions, ($\mathbf{q}(0), \mathbf{K}(0)$).

4. SOLVING THE MODEL

4.1. Programming the Solution

To program the exercise, software components are needed to solve differential equations and undertake searches for a range of parameter sets. We used Matlab (Mathworks, 2002) as it is ideally suited for this type of computational problem. The programming was written so as to make use of key Matlab features. Library routines (toolboxes) were used so that start-of-the art solvers and searches are included in the code. Using the extensive matrix capabilities allowed for exactly the same code being executed for all dimensionalities greater than one.

All results were generated using Matlab 6.1 on the same computer: a Dell Latitude Notebook with Pentium 3 running at 1.3 GHz and 256Mb of RAM.

4.2. Parameter Calibration

To generate the results presented here, we repeatedly solved the model over a range of parameter sets. A total of 100 model repetitions are used for each dimensionality, m (=2n). Each model repetition differs only in the parameter calibration. For all models $a = 1, r_0 = 0.03$ and $r = 0.05$, and the models differ because of the choice of α_i's and b_i's which are chosen from the following distributions:

$$\alpha_i = \frac{1}{3n} + \frac{\xi\eta_i}{2\sum\eta_i}, \; i = 1, 2, ..., n \qquad (22)$$

$$b_i = 3 + 4\delta_i, \; i = 1, 2, ..., n \qquad (23)$$

where ξ, η_i, δ_i are each drawn from $U(0,1)$, the random uniform distribution between 0 and 1. The α_i's and b_i's determine the extent of model non-linearity. Hence, by employing a range of values as given by equations (22-23), we are able to investigate the average properties of a broad range of nonlinear models.

This choice in parameter sets produces a suite of model repetitions that have a sensible economic meaning and that are reasonably well behaved from a computational perspective, yet give a wide-ranging parameter space.

4.3. Timing Experiment

Calibrating the model and applying suitable stopping rules we were able to conduct timing experiments as the dimensionality of the model is allowed to increase from 2 to 40. These experiments were designed to evaluate how long each algorithm takes to complete, irrespective of whether the algorithm completed successfully or unsuccessfully. The results of these timing experiments are summarized in Figure 2.

From the theoretical proposition derived in Section 2, we observed that the reverse-shooting algorithm is more efficient than the forward-shooting algorithm in the sense that the reverse-shooting algorithm requires searching over a smaller dimensional space. This proposition is supported by the results summarized in Figure 2, where it is shown that, as the dimensionality of

the model increases, the time taken to solve the forward-shooting algorithm grows at an exponentially greater rate than does the time taken to solve the reverse-shooting algorithm

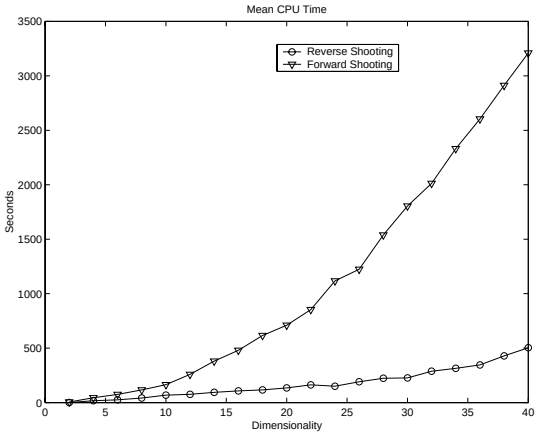

Figure 2. Mean CPU Time to Solve Model Using Alternative Algorithms

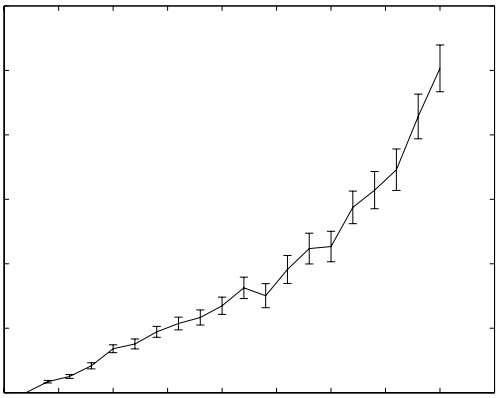

Figure 3. Reverse-Shooting: Mean Plus Two Standard Deviations

The mean CPU times have been derived by calculating the mean of 100 replications for each dimensionality. Accordingly, it is also possible to calculate two standard deviation confidence intervals for the mean times. Of course, the number of replications has a major influence on the magnitude of the standard deviations. In particular, increasing the number of replications will reduce the size of the confidence intervals. Figures 3 and 4 demonstrate that, for reverse-shooting and forward-shooting, the sizes of the confidence intervals increase with the dimensionality of the model; hence the confidence intervals also increase with mean CPU time.

5. MEASURING SUCCESS

5.1. Success Rates of Competing Algorithms

There are a number of important computational issues that will affect the solution to this problem, which is very sensitive to a whole range of approximations that are made in the solution process. Firstly, there is the parameter space. The model will be reasonably well-behaved computationally as it is an economic problem (and, thus, for example, cannot have negative capital stocks). But the parameter space will affect the size (though not the dimensionality) of the solution space. Secondly there is the choice of the differential equation solver and thus the truncation errors and ability to handle different speeds in the solution dynamics. Thirdly there is the method of searching over the candidate solution trajectories. Finally there are the definitions of "close enough" in both the solver and in the search. All these issues combine in producing errors and in producing the solution. All may increase over wider parameter spaces and dimensionalities.

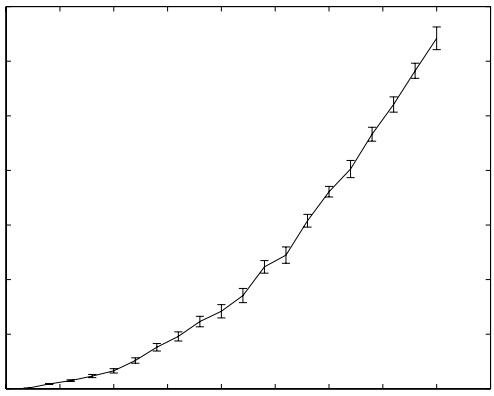

Figure 4. Forward-Shooting: Mean Plus Two Standard Deviations

All these factors will compound to determine success or failure of each algorithm in any particular case. It is possible that one algorithm may be more successful than another. In evaluating the effectiveness of each algorithm, it is also important to take success into account, in addition to measuring CPU time. Calculation of success rates would be an important extension of this study.

5.2. Econometric Modeling

Econometric techniques can be used to summarize the likely success of alternative approaches when confronted by models with differing properties.

We could consider the success or failure of each algorithm for each given dimensionality, over a distribution of parameter values. We would then use the distribution of successes and failures to estimate a multivariate probit model where the probability of each algorithm being successful depends on the dimension of the model, it's degree of nonlinearity, the size of the parameter space, and the values of key parameters.

Suppose we are considering two algorithms, the forward-shooting and the reverse-shooting algorithms. Let y_{1i} be a binary variable equal to 1 when the forward-shooting algorithm is successful for solving the i-th model configuration, and 0 when it is unsuccessful. Let y_{2i} be a similar variable for describing the success of the reverse-shooting algorithm. The i-th model configuration is described by a vector $\mathbf{x}_i$, that includes the dimension of the model and various parameter settings.

In the probit model formulation, the probability that the j-th algorithm is successful for solving the i-th model configuration is given by

$$\Pr(y_{ji} = 1) = \Phi(\mathbf{x}_i^T \boldsymbol{\beta}_j) \qquad (24)$$

where $\Phi(.)$ is the distribution function for a standard normal distribution, and the $\boldsymbol{\beta}_j$ are vectors of unknown parameters. Noting that the success of one algorithm is unlikely to be independent of the success of the other algorithm, we write the probability that both algorithms are successful as

$$\Pr[(y_{1i} = 1) \cap (y_{2i} = 1)] = \Phi_2(\mathbf{x}_i^T \boldsymbol{\beta}_1, \mathbf{x}_i^T \boldsymbol{\beta}_2, \rho) \qquad (25)$$

where $\Phi_2(.,.,\rho)$ is the distribution function for a bivariate standard normal distribution with correlation parameter ρ. Given the success or otherwise for each algorithm under a large number of settings for $\mathbf{x}_i$, we can proceed to estimate β_1, β_2 and ρ.

In future studies, we plan to use Bayesian inference along the lines described by Chib and Greenberg (1998). In addition to estimating the probability of success for each (and both) algorithms for a range of settings of $\mathbf{x}_i$, using Bayesian inference allows us to express uncertainty about these probabilities and to find probability distributions for the ranking of the algorithms in terms of their likely success. It will be possible to consider how factors such as the nonlinearity of the models, the size of the parameter space and the dimensionality of the problem influence the success rate of the competing algorithms. This will enable us to evaluate how the likely success rate changes as the dimensionality of the model increases and as the extent of non-linearity of the model increases.

6. CONCLUDING REMARKS

This paper has used two well-known algorithms to solve a multi-dimensional investment model. We have examined the CPU time taken by the two algorithms as the dimensionality of the model is allowed to increase. We have suggested theoretically, and also demonstrated computationally, that the reverse-shooting algorithm can be significantly faster than the forward-shooting algorithm.

A more complete analysis of this topic would take into account the fact that the two algorithms may have significantly different success rates. The penultimate Section of this paper has described an econometric framework where this question could be addressed in a systematic manner.

7. ACKNOWLEDGMENTS

Ric Herbert is grateful to The University of Newcastle for financial assistance through an Early Career Research Grant. Peter Stemp and Bill Griffiths are grateful to the Faculty of Economics and Commerce at The University of Melbourne for financial assistance through Faculty Research Grants.

8. REFERENCES

Blanchard, O. J., and C. M. Kahn (1980), "The Solution of Linear Difference Models under Rational Expectations," Econometrica, 48, pp. 1305-1311.

Chib, S. and E. Greenberg (1998), "Analysis of Multivariate Probit Models", *Biometrika*, 85, 347-361.

Hayashi, F. (1982), "Tobin's Marginal q and Average q: a Neoclassical Interpretation," Econometrica, 50, pp. 213-224.

MathWorks (2003), http://www.mathworks.com/