ISSN 0819-2642
ISBN 0 7340 2513 0

THE UNIVERSITY OF MELBOURNE

DEPARTMENT OF ECONOMICS

RESEARCH PAPER NUMBER 858

SEPTEMBER 2002

REVERSE SHOOTING IN A MULTI-DIMENSIONAL SETTING

by

Ric D. Herbert
&
Peter J. Stemp

Department of Economics
The University of Melbourne
Melbourne Victoria 3010
Australia.

REVERSE SHOOTING IN A MULTI-DIMENSIONAL SETTING[#]

by

Ric D. Herbert[*] and Peter J. Stemp[**]

ABSTRACT

This paper investigates the properties of dynamic solutions that have been derived using the well-known reverse-shooting algorithm. Given an arbitrary large-scale model about which we have limited information, how successful is the algorithm likely to be in solving this model? We address this question using a range of investment models, both linear and non-linear. By extending the investment models to allow for multi-dimensional specifications of the capital stock, we are able to examine the computational efficiency of the reverse shooting algorithm as the dimensionality of the capital stock is allowed to increase. Our approach provides insights into how the complexity of the solutions to a broad range of macroeconomic models increases with the dimensionality of the models.

JEL classification: C63; E27

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

[#] An earlier version of this paper was presented to the 8th International Conference of the Society for Computational Economics, Computing in Economics and Finance, Aix-en-Provence, France. June 27-29, 2002.

[*] Faculty of Science and Information Technology, The University of Newcastle – Ourimbah, Ourimbah, NSW, 2258, Australia. *email: ric.herbert@newcastle.edu.au*

[**] Department of Economics, The University of Melbourne, Melbourne, Victoria, 3010, Australia, email: pjstemp@unimelb.edu.au. Peter Stemp is grateful to The University of Melbourne for financial support through a Melbourne Research Development Grant and a Faculty Research Grant.

1. INTRODUCTION

Given an arbitrary large-scale model about which we have limited information, how successful is the well-known reverse shooting algorithm likely to be in solving this model? We address this question using a series of investment models with specific properties that are common to a range of macroeconomic models. Firstly, the chosen models are derived from an optimising framework.

Secondly, the models have a number of stable and unstable trajectories so that it is likely to be complicated to solve each model for a stable solution. The economy is initially at a stable steady state equilibrium, and when shocked by, say, an exogenous change in interest rates, then it moves to a stable trajectory leading to a new steady state equilibrium. The movement to the new equilibrium is assumed to come about as a consequence of optimising behavior of the agents in the model. In each of the models, certain variables jump instantaneously after the shock, and force the model dynamics onto the trajectory leading to the stable equilibrium.

A third property of the models is that they are nonlinear with nonlinearities arising as a direct consequence of optimising behavior. The usual approach is to linearise each model in the neighborhood of the steady state and then to solve the linearised model. Of course, it is always possible to find closed-form solutions for the linearised models using matrix techniques. Such matrix solutions are likely to be more computationally efficient than solutions derived using a search algorithm. However, the solution properties derived by applying the reverse shooting algorithm to the linearised models are also going to give an informative benchmark by providing an indication of how successful the algorithm is likely to be in solving an arbitrary large-scale model that is "almost" linear.

The essential issue is that of first finding the stable manifold and then finding a unique stable path along this manifold that gives the dynamic solution. In the case where there is one stable and one unstable eigenvalue the stable manifold is a (one-dimensional) path in two dimensions and this path is the stable solution. In the higher-dimensional case (with more than one stable eigenvalue) the stable manifold has dimensionality greater than one and the stable path is a sub-set of the stable manifold. Appropriate jumps in the variables ensure that the model solution is on the stable path. This issue has been considered, especially in the case of rational expectations variables, by a number of authors including Anderson and Moore (1985), Blanchard and Kahn (1980), Boucekkine (1995), Fair and Taylor (1983), Judd (1998) and Zadrozny (1998).

The basic computational problem that we investigate is how well the reverse shooting approach solves the example problem over a range of parameter spaces, dimensionalities and computational parameters. We are particularly interested in what is commonly referred to as Bellman's curse of dimensionality, in that we wish to investigate to what extent the computational effort required for solving the problem increases with dimensionality.

In particular, we investigate the computational effort needed to solve the dynamics of the both linear and non-linear models as the numbers of stable and unstable eigenvalues are allowed to increase. We demonstrate that this means that the dimensionality of the stable manifold and the computational complexity of the problem will increase.

2. SADDLE-PATH INSTABILITY: THE TWO-DIMENSIONAL PROBLEM

Description of saddle-path

Consider the following two-dimensional model:

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

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

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \ \mathbf{x} = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \text{ and } \mathbf{x}^* = \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} \tag{1b}$$

Assume that $\mathbf{A}$ 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 $\mathbf{A}$ associated with λ_i so that:

$$\mathbf{v}(\lambda_1) = \begin{bmatrix} a_{12} \\ \lambda_1 - a_{11} \end{bmatrix} \text{ and } \mathbf{v}(\lambda_2) = \begin{bmatrix} a_{12} \\ \lambda_2 - a_{11} \end{bmatrix} \tag{1c}$$

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

$$\mathbf{x}(t) = \begin{bmatrix} a_{12} & a_{12} \\ \lambda_1 - a_{11} & \lambda_2 - a_{11} \end{bmatrix} \begin{bmatrix} C_1 \exp(\lambda_1 t) \\ C_2 \exp(\lambda_2 t) \end{bmatrix} \tag{1d}$$

This solution has the property of saddle-path instability, which will be described more fully using the investment example below.

Example

As an example, consider the investment decision of a profit maximising firm which takes the supply of labour as given. 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 a parameter, b. The decision of the firm can then be summarised as follows:

Choose I and W to maximise:

$$V = \int_0^\infty e^{-rt}[F(K,L) - WL - I]dt \tag{2a}$$

subject to

$$\dot{K} = I - b\left(\frac{I^2}{K}\right) \tag{2b}$$

$$F(K,L) = aK^{\alpha}L^{1-\alpha} \tag{2c}$$

where

K = real stock of capital
L = supply of labour (assumed exogenous)
W = real wage
I = real level of investment
$F(K, L)$ = real output
r = real interest rate (assumed exogenous)
a, b are exogenous parameters

If the supply of labour L is fixed at unity and the real wage is fixed at its optimal level then the dynamics of capital accumulation in the model reduce to the following set of equations:

$$\dot{q} = [r - b(\Phi(q))^2]q - F_K \tag{3a}$$

$$\dot{K} = \Phi(q)[1 - b\Phi(q)]K \tag{3b}$$

where

$$F_K = a\alpha K^{\alpha-1} \tag{3c}$$

$$\Phi(q) = \frac{q-1}{2bq} \tag{3d}$$

The variable q is the co-state variable derived from the firm's optimisation problem. Given the functional forms of investment and production functions encapsulated in equations (2b, 2c), Hayashi (1982) demonstrates that q is equal to (both average and marginal) Tobin's q.

The model defined by equations (3a-3d) is non-linear. However, we can obtain a general idea about the stability properties of the model by linearising in the neighbourhood of the steady state. The linearised model is given by:

$$\begin{bmatrix} \dot{q} \\ \dot{K} \end{bmatrix} = \begin{bmatrix} r & -F_{KK} \\ \dfrac{K^*}{2b} & 0 \end{bmatrix} \begin{bmatrix} q - q^* \\ K - K^* \end{bmatrix} \tag{4a}$$

where

$$F_{KK} = a\alpha(\alpha - 1)\left(K^*\right)^{\alpha-2} < 0 \tag{4b}$$

The eigenvalues of the linearised system are given by:

$$\lambda_1, \lambda_2 = \frac{r \pm \sqrt{r^2 - \dfrac{2F_{KK}K^*}{b}}}{2} \tag{4c}$$

Hence the linearised system has two real-valued eigenvalues given by $\lambda_1 < 0, \lambda_2 > r$ and exhibits the property of saddle-path instability.

Solutions to the two-dimensional model starting from a range of initial conditions can be used to derive a phase diagram for the dynamics of the two-dimensional model. Phase diagrams for the true (nonlinear) and linearised models are given in Figures 1 and 2, respectively. The same parameter set is used in both cases, so the isoclines in the two figures are directly comparable. The figures show saddle-path dynamics of the two-dimensional model. They also show that there are substantial differences in the dynamics between the nonlinear and linearised models.

(Figures 1 and 2 about here)

Solving the two-dimensional model is then equivalent to solving the following problem.

Find q(0) subject to:

$$\dot{q} = f(q, K) \tag{5a}$$

$$\dot{K} = g(q, K) \tag{5b}$$

$$K(0) = K_0^* + \mu_K \tag{5c}$$

$$q(\tau) = q^* + \varepsilon_q \tag{5d}$$

$$K(\tau) = K^* + \varepsilon_K \tag{5e}$$

where τ is some (possibly exogenous, possibly endogenous) large number representing the terminal point for time and ε_q, ε_K and μ_K are small error terms that are "close enough" to zero.

The computational problem

Solving the computational problem in the two-dimensional case is relatively easy with reverse shooting. The aim of this approach is to find the stable trajectories of the model and generate the stable arms in *q*-*K* phase space. This approach makes use of the feature that time can be abstracted from the solution of the model. The stable arms forwards in time will become the unstable arms with time going backwards. The same will apply for the unstable arms, with reverse time making them the stable arms. This approach finds the forward-stable arms by finding the unstable arms in reverse time (backward-unstable arms). This motivates the word reverse in the name for the approach.

The approach also makes use of the separatrix property of saddles (Khalil, 1996). The stable trajectories from a saddle form a separatrix so that the phase plane of the model is divided into four separate regions. Solutions always remain in one and only one region. Choosing a solution close to the boundary of one of these regions will ensure that the solution will remain close to the boundary. Choosing a backward-unstable solution close to the boundary will provide a time-path for the forward-stable solution (stable arm).

We use a differential equation solver for the model and start near enough to the steady state given by $\mathbf{x}^* = (K^*, q^*)$, so there is some transient dynamics. We then solve the model in reverse time and the solution will be forced onto the stable arm. We stop the solver when the solver generates a solution that is "close enough" to the initial conditions for the capital stock, and the resulting solution gives the initial conditions for the co-state.

(Figure 3 about here)

Figure 3 shows a stable arm for the linearised and the true (nonlinear) model. The figure shows the dynamics of the model in response to an interest rate shock from $r_0 = 0.03$ to $r = 0.05$. The stable arms have been derived using the reverse shooting approach. Once the stable arm (or forward-stable trajectory) has been determined in this manner, the initial value for q can be obtained by reading the corresponding value of $q(0)$ along the stable arm for the initial condition $K(0)$.

Using this technique we can find a computational solution to the problem. The two-dimensional model can always be solved with only one pass by the solver. The resultant solution is not the "true" solution as the solver introduces truncation errors and round-off errors will be introduced through the use of floating point numbers. This can result in substantial errors due to the unstable nature of this problem, for the problem is not well posed, in the sense that a slight change to the initial conditions is likely to lead to substantial differences in the final solution. The reason for this is that there are saddle-path properties inherent in the model solution.

One way to check the solution is to solve the model in forward time from the initial conditions that have been discovered using the reverse shooting approach and check that the resulting forward trajectory gets "close enough" to the steady state.

This gives more confidence in the solution, but still not the "true" solution. This raises the issue of how good needs to be the solution and how close is "close enough"? We consider these computational parameter issues in out numeric results presented below.

3. THE HIGHER DIMENSIONAL PROBLEM

General problem

Consider the following model:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\mathbf{x}(t) - \mathbf{x}^*) \qquad (6)$$

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 $\underset{mx1}{\mathrm{v}(\lambda_i)}$ 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:

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

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 at the beginning of the optimisation problem 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 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 $i = 1, 2, ..., s$.

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:

$$\mathbf{z}(t) - \mathbf{z}^* = \begin{bmatrix} \mathbf{v}(\lambda_1) & \mathbf{v}(\lambda_2) & \cdots & \mathbf{v}(\lambda_s) & \mathbf{v}(\lambda_{s+1}) & \mathbf{v}(\lambda_{s+2}) & \cdots & \mathbf{v}(\lambda_{s+u}) \end{bmatrix} \begin{bmatrix} C_1 \exp(-\lambda_1 t) \\ C_2 \exp(-\lambda_2 t) \\ \vdots \\ C_s \exp(-\lambda_s t) \\ C_{s+1} \exp(-\lambda_{s+1} t) \\ C_{s+2} \exp(-\lambda_{s+2} t) \\ \vdots \\ C_{s+u} \exp(-\lambda_{s+u} t) \end{bmatrix} \tag{8}$$

Without loss of generality, start at $\tau = -N$ where N is a large positive number and choose $\mathbf{z}(-\tau)$ 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 (3) reduces to:

$$\mathbf{z}(t) - \mathbf{z}^* = \mathbf{x}(-t) - \mathbf{x}^* = \begin{bmatrix} \mathbf{v}(\lambda_1) & \mathbf{v}(\lambda_2) & \cdots & \mathbf{v}(\lambda_s) \end{bmatrix} \begin{bmatrix} C_1 \exp(-\lambda_1 t) \\ C_2 \exp(-\lambda_2 t) \\ \vdots \\ C_s \exp(-\lambda_s t) \end{bmatrix} \tag{9}$$

A solution can then be found by searching over the values for $(C_1, C_2, ..., C_s)$ until a solution is found that arrives within a suitably small neighbourhood 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.

When the dimensionality is greater than two, the solution method for the problem is similar but more complicated than the two-dimensional case. In the two-dimensional problem, the stable manifold is a one-dimensional line, which uniquely defines the stable path. In the higher-dimensional problem, the stable manifold is at least two-dimensional so that it contains an infinite number of one-dimensional paths. In order to find the stable path it is necessary to search over the stable manifold for the "right" solution path.

Visualising the stable manifold

The simplest example where the stable manifold is not the same as the stable path occurs in the situation where there is one unstable eigenvalue and two stable eigenvalues. Figure 4 is drawn under the assumption that there are two variables which are predetermined by history, K_1 and K_2, and one "jump" variable, q_1. Then, reverse shooting is equivalent to searching over a two-dimensional space (the stable manifold) in the neighbourhoood of the steady state for the unique path that passes through the values, K_{10} and K_{20}. When this path has been determined, the intersection of K_{10} and K_{20} with the stable manifold also determines the initial value for the "jump" variable, q_1.

(Figure 4 about here)

The stable path is found by initiating the solver at a point that is close to the steady state and then solving the dynamics in reverse time. The resultant solution path will lie on the stable manifold but may not pass through the desired values, K_{10} and K_{20}. If not, it is necessary to again initialise the solver close to the steady state and to solve the problem again. A systematic search, with different initial values for the solver in the neighbourhood of the steady state, may eventually converge to a solution that passes "close enough" to (K_{10}, K_{20}).

Example of higher dimensional problem

We next examine a special example of dimension *2n* where $s = u = n$. This is the example that we will focus on during the rest of the paper. Consider the investment decision of a profit maximising 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 summarised as follows:

Choose the I_i to maximise:

$$V = \int_0^\infty e^{-rt}[F(K_1, K_2, ..., K_n) - \sum_{i=1}^{n} I_i]dt \qquad (10)$$

subject to

$$\dot{K}_i = I_i - b_i\left(\frac{I_i^2}{K_i}\right), \text{ for } i = 1, 2, ..., n \qquad (11a)$$

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

$$F(K_1, K_2, ..., K_n) = a\prod_{i=1}^{n}(K_i)^{\alpha_i}, \text{ where } \sum_{i=1}^{n}\alpha_i < 1 \qquad (11c)$$

where

K_i = real stock of capital of type 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:

$$\dot{q}_i = [r - b_i(\Lambda(b_i, q_i))^2]q_i - F_i, \text{ for } i = 1, 2, ..., n \tag{12a}$$

$$\dot{K}_i = \Lambda(b_i, q_i)[1 - b_i\Lambda(b_i, q_i)]K_i, \text{ for } i = 1, 2, ..., n \tag{12b}$$

where

$$F_i = F_{K_i} = \frac{a\alpha_i}{K_i}\prod_{i=1}^{n}(K_i)^{\alpha_i} \tag{12c}$$

$$\Lambda(b_i, q_i) = \frac{q_i - 1}{2b_i q_i} \tag{12d}$$

The variables q_i are the co-state variables derived from the firm's optimisation 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 (where an asterisk denotes the steady state value):

$$q_i^* = 1, \text{ for } i = 1, 2, ..., n \tag{13a}$$

$$K_i^* = \left[\frac{r}{\alpha_i a}\prod_{j \neq i}\left(\frac{\alpha_i}{\alpha_j}\right)^{\alpha_j}\right]^{\frac{1}{\sum_i \alpha_i - 1}}, \text{ for } i = 1, 2, ..., n \tag{13b}$$

The model defined by equations (12a-12d) is non-linear. However, we can obtain a general idea about the dynamic properties of the model by linearising in the neighbourhood of the steady state. The linearised model is given by:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \\ \dot{K}_1 \\ \dot{K}_2 \\ \vdots \\ \dot{K}_n \end{bmatrix} = \begin{bmatrix} r & 0 & \cdots & 0 & -F_{11} & -F_{12} & \cdots & -F_{1n} \\ 0 & r & \cdots & 0 & -F_{21} & -F_{22} & \cdots & -F_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r & -F_{n1} & -F_{n2} & \cdots & -F_{nn} \\ \frac{K_1^*}{2b_1} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \frac{K_2^*}{2b_2} & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{K_n^*}{2b_n} & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} q_1 - 1 \\ q_2 - 1 \\ \vdots \\ q_n - 1 \\ K_1 - K_1^* \\ K_2 - K_2^* \\ \vdots \\ K_n - K_n^* \end{bmatrix} \tag{14}$$

where $F_{ij} = F_{K_i K_j}$.

In addition, for the linearised model, the following second-order conditions for profit maximisation are satisfied:

$$K_{ii} < 0 \text{, for } i = 1, 2, \ldots, n \tag{15a}$$

$$(-1)^n \det H_n > 0 \tag{15b}$$

where

$$H_n = \begin{bmatrix} F_{11} & F_{12} & \cdots & F_{1n} \\ F_{21} & F_{22} & \cdots & F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ F_{n1} & F_{n2} & \cdots & F_{nn} \end{bmatrix}. \tag{15c}$$

Whenever equation (15b) is satisfied, it can be shown that the model given by equation (14) has precisely n stable and n unstable eigenvalues and so fits the structure of the general model presented above. In particular, any production technology of the form given by equation (11c) satisfies equations (15a-15c).

The computational problem

The computational problem we examine is the solution of the example 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 systems:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}) \tag{16a}$$

for the non-linear model, and

$$\dot{\mathbf{x}}_{\mathrm{L}} = \mathbf{A}(\mathbf{p})(\mathbf{x}_{\mathrm{L}}(t) - \mathbf{x}^*) \tag{16b}$$

for the linear model, where the state vector is given by:

$$\mathbf{x} = [\mathbf{q}, \mathbf{K}]^T = [q_1, q_2, ..., q_n, K_1, K_2, ..., K_n]^T \tag{16c}$$

and the parameter vector is given by:

$$\mathbf{p} = [a, \alpha_1, \alpha_2, ..., \alpha_n, b_1, b_2, ..., b_n]^T \tag{16d}$$

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 $\mathbf{x}_0^* = [\mathbf{q}_0^*, \mathbf{K}_0^*]^T$ and evolves along a unique stable solution trajectory to $\mathbf{x}^* = \left[\mathbf{q}^*, \mathbf{K}^*\right]^T$. From equations (16a-16b), both these steady state equilibria can be analytically determined. The problem is to find this stable trajectory. The interesting aspect of the problem results from the model structure in that 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. This is a similar problem to that defined by equations (5a-5e).

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, but one thing in our favour is that the final steady state is known, as given by equations (13a-13b). Basically we need to search around points "near enough" to the final steady state so that the solution to the model has transient dynamics that are forced onto the stable manifold and that also satisfy the appropriate inital conditions. This search will determine a solution trajectory and initial conditions, $(\mathbf{q}(0), \mathbf{K}(0))$, as in the two-dimensional case. After a solution path has been derived using the reverse shooting algorithm, solving the model forwards in time using these initial conditions can check the accuracy of the solution by examining how close the resulting trajectory is to reaching the steady state.

4. PROGRAMMING THE SOLUTION

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.

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.

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:

$$\alpha_i = \frac{1}{3n} + \frac{\xi\eta_i}{2\sum\eta_i},\ i = 1, 2, \dots, n \tag{17a}$$

$$b_i = 3 + 4\delta_i,\ i = 1, 2, \dots, n \tag{17b}$$

where ξ, η_i, δ_i are each drawn from $U(0,1)$, the random uniform distribution between 0 and 1. Note that, for the nonlinear models, the α_i's and b_i's determine the extent of model nonlinearities. Hence, by employing a range of values as given by equations (17a-17b), 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 computationally, yet give a wide-ranging parameter space. In particular, the interaction between the parameter values and the definition of the steady state value of the capital stock given by equation (13b) means that, for low dimensions, the size of the search space within that dimensionality is much larger than is the case for higher dimensions. This proposition is illustrated in Table 1.

(Table 1 about here)

ODE solver software

Solving this computational exercise is all about finding the final solution trajectory for a given model as determined by a parameter set. This final solution trajectory is a single solution to an ordinary differential equation. It is simply the solution to equation (12a-12b) from the correct set of initial conditions or equation (14) in the case of the linearised model. To find this solution trajectory it is often necessary to solve thousands of ordinary differential equations. We refer to each solution of a differential equation as a candidate solution.

Basically the higher-dimensional reverse shooting problem comes down to solving many differential equations. The choice of the software component to solve the ordinary differential equations in this exercise will have considerable implications for the results. Small changes to the initial conditions of the ordinary differential equation will lead to huge differences in the final solution. For the differential equation solver we use a variable time step size Runge-Kutta method solver. This is a well-known and standard ode solver for this type of problem. It has the key features of robustness and accuracy, and it can cope with the problem "blowing-up". It is well suited to the type of dynamics generated by the examples chosen in this paper.

We implement the solver by calling the Matlab function *ode45*. The time step is chosen so that the local truncation error is less than 10^{-4}. We use a long time horizon (ranging from 0 to 1500) but use the "events" property of the Matlab ode solver suite to stop the integration of a candidate solution so that only a small fraction of the time horizon is normally used. This, of course, significantly reduces the computational effort needed to solve the exercise. The resulting time horizon will be variable with each candidate solution. As an example of a solver stopping condition, a candidate reverse time trajectory is stopped as soon as any capital stock is greater than its corresponding initial steady state. This candidate can then be abandoned. The "greater than" comes from the fact that, for $r_0 = 0.03$ and $r = 0.05$, $\mathbf{K}^* < \mathbf{K}_0^*$.

Searcher software

Solving this computational exercise involves searching over many candidate solution trajectories to find the "correct" trajectory. Recall that for a reverse-shoot candidate solution to be the "correct" solution all capital stocks must pass sufficiently closely to the initial steady state at the same time. From this "correct" solution comes the initial conditions required to solve the model and thus the jumps in $\mathbf{q}$. The searcher software generates candidate solutions and stops when it finds the "correct" candidate.

For the reverse shoot, a candidate ordinary differential equation is solved in reverse time from a set of terminal conditions close to the final steady state. Effectively the searcher software generates these terminal conditions. The searcher's software sits over the top of the ode solver software, and generates solutions until it finds the "correct" solution. Thus the choice of the searcher software component is also important for the solution of the exercise.

For the search method we use a Nelder-Meade direct simplex search. This search has the advantage that it has memory and can go back to previous search candidates (simplex vertices) and thus is less likely to get "stuck" in a search. Unlike many other search procedures it does not require the generation (by analytic or numeric means) of derivatives. Like most searches, it works best at low dimensionalities (Lararias et al., 1998). We have found it to be a good robust searcher for this type of problem compared to other searchers we have used.

We implement the Nelder-Meade search by the Matlab function "*fminsearch*" from the Optimization Toolbox. The software is implemented by defining an objective function that is to be minimised. Like all searcher software, this function has a number of stopping conditions. These include that the objective function reaches a minimum as defined by a tolerance and within a maximum number of iterations. Alternatively, successive iterates may differ by less than a specified tolerance. Note that successful searches do not mean that the global minimum has been found. The tolerances are chosen as computational parameters. We consider their effects on the solving of this exercise by considering two tolerances: the lower tolerance is 0.0001; the higher tolerance is 0.1.

The search combines two aspects in trying to find the candidate that is the stable trajectory. Firstly it needs to stay close to the final steady state so that this candidate's "initial" conditions for the reverse time solution has transient dynamics that are forced onto the stable manifold. These "initial" conditions to the candidate ode are actually terminal conditions for the original model. If they are too close to the steady state then there will be no transient dynamics as the model will simply remain at the steady state. Secondly the search needs to find the candidate solution trajectory that passes close enough to the pre-shock steady state capital stocks, $\mathbf{K}_0^*$. All capital stocks must be close enough at the same time point.

For the objective function of the search we use the 2-norm of the relative error of each these two components and sum the result. Hence there is a "trade-off" between each component. We also weight the components by dividing by $\sqrt{n}$, where m (=$2n$) is the dimensionality of the problem. This is to allow for the effects of parameter generation rules with increase in dimensionality. That is the search is chosen so as to minimise:

$$J = \frac{1}{\sqrt{n}}\left(\left(\frac{K_1(0)-K_{01}^*}{K_{01}^*}\right)^2 + \left(\frac{K_2(0)-K_{02}^*}{K_{02}^*}\right)^2 + \ldots + \left(\frac{K_n(0)-K_{0n}^*}{K_{0n}^*}\right)^2\right)^{\frac{1}{2}}$$

$$+\frac{1}{\sqrt{n}}\left(\left(\frac{q_1(\tau)-q_1^*}{q_1^*}\right)^2 + \ldots + \left(\frac{q_2(\tau)-q_2^*}{q_2^*}\right)^2 + \left(\frac{K_1(\tau)-K_1^*}{K_1^*}\right)^2 + \ldots + \left(\frac{K_n(\tau)-K_n^*}{K_n^*}\right)^2\right)^{\frac{1}{2}} \tag{18}$$

Successful solution

If the search terminates successfully, we may or may not have the "correct" candidate solution. The search may terminate successfully as this is the correct candidate or because it could not find a better one. From this "correct" solution comes the initial condition that gives the "true" solution trajectory and the jumps in $\mathbf{q}$. For the purposes here, we can set up the ultimate test of the method by solving the model forwards in time from the initial conditions to check that the model solves to the steady state. How close forward trajectory from the initial conditions found by reverse shooting is to the steady state is the ultimate test of the reverse shooting procedure for this exercise.

For the forward solutions we define a normalized forward trajectory error as a measure of "how close" the solution came to the desired steady state. The normalized errors are measured as relative errors using 2-norms, and are normalized to search space and dimensionality by dividing by the maximum capital stock and the square root of the dimensionality. This normalization allows for the wide difference in search spaces generated by the parameter generation rule (equations 17a-17b).

Thus, successfully solving the model involves both the completion of a successful search using the reverse shooting algorithm and the successful verification of the search by checking that the corresponding forward trajectory passes "close enough" to the final steady state. For our purposes a successful reverse shoot search is defined by the tolerance parameter (in our paper, equal to 0.0001 or 0.1). A successful forward solution is defined as one with normalised forward trajectory error < 0.1.

5. RESULTS

Simulations were implemented for dimensionalities of 4 to 40, where m ($=2n$) is the dimensionality of the model. For our investment model, n is the dimensionality of the stable manifold. The two-dimensional model has not been included for presentation reasons. As this model does not require a search, it is significantly faster and has less variability in solution time. All results were generated using the same computer[1].

[1] Matlab 6.1 on a Dell Latitude Notebook with Pentium 3 running at 1.3 GHz and 256Mb of RAM.

CPU time to solve models

The first issue we examine is the effects of dimensionality on the time it takes to solve a model. From Figure 5 it can be seen that the linear model is significantly faster to solve than the non-linear model. The time and variability of the model solutions also increases monotonically with dimensionality. This is especially apparent with the non-linear model where variability and solution time increases more rapidly with dimensionality than for the linear model.

(Figure 5 about here)

Success rate

Figure 6 presents the results for successful solving the model, that is, for successfully completing a search and then successfully checking the forward trajectory. The linear model solves best (more than 80% success rate) for dimensionalities ranging between 8 and 22. The non-linear model has above 80% success rate in only one case (dimensionality of 6) and less than 50% success rate for all dimensionalities greater than 12.

(Figure 6 about here)

Clearly the reverse shooting algorithm is struggling to solve the nonlinear models even for very low dimensionalities. We suspect that the algorithm has difficulties solving the linear model for two different reasons. At low dimensionalities (4 and 6) we suspect that this is because of the large search space generated by the parameters (and described in Table 1) interacting with the search algorithm where successful searches terminated with close iterates rather than close to the global minimum. At high dimensionalities (24 and greater) we suspect this is because of the higher dimensionality.

Varying search tolerance

We have also examined outcomes when the search tolerance is allowed to increase from 0.0001 to 0.1. The outcomes for CPU time and success rate are described in Figure 7. As in Figure 6, the success rate figures reported here are for successfully solving the model (that is, successful search plus successful forward trajectory).

(Figures 7 and 8 about here)

As one would expect the increased tolerance means that the search can be completed sucessfully in lesser time. More interesting is the observation that the success rate for the linear model tends to improve substantially for lower dimensions under the higher search tolerance. This proposition is demonstrated further in Figure 8 and is consistent with the earlier observation that the problem with lower success rates at low dimensionalities is because of the interaction between the stopping rule for the search algorithm and size of the search space. Overall, these results suggest that there are gains to be made in choosing a higher tolerance level for low dimensions of the linear model

Overview of results

The reverse shooting algorithm is clearly a lot faster for linear models than it is for nonlinear models. Other things being equal, the probability of achieving a successful solution with this algorithm decreases with the following factors:

- reductions in the linearity of the model;
- increases in the dimensionality of the model for given search space; and,
- increases in the search space for given dimensionality.

Overall, our results suggest that the algorithm does best with "almost" linear models of dimensionality less than about 20. Our results also suggest that the success of the algorithm can be improved somewhat for linear models by increasing the search tolerance for lower dimensionalities (less than 10). For the nonlinear models considered here there is substantial evidence that the algorithm is prone to failure at dimensionality around 10 or greater with less than 50% success rate for all dimensionalities greater than 12.

6. CONCLUSION

In this paper we apply the reverse shooting algorithm to solving a range of investment models, both linear and non-linear. By extending the investment models to allow for multi-dimensional specifications of the capital stock, we are able to examine the computational efficiency of the reverse shooting algorithm as the dimensionality of the capital stock is allowed to increase.

We investigated the success of the reverse shooting algorithm for models ranging in dimensionality between 4 and 40, discovering that the algorithm had considerable problems solving this type of exercise. Reverse shooting does not work well at high dimensionalities, achieving best results for linear models with dimensionality below 20. Over the range of dimensionalities considered, effort to successfully solve the model increases monotonically with dimensionality with the non-linear model requiring more effort and having greater variability in effort than the linear model. Our results indicate that even the introduction of "well-behaved" linearities like those introduced in this paper, can substantially reduce the effectiveness of the reverse shooting algorithm.

The exercise was 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 soon "blow-up". It is a good exercise on which to test the appropriateness of the reverse shooting method.

Our approach has provided insights into how the complexity of the solutions to a broad range of macroeconomic models increases with the dimensionality of the models. The results raise the question as to whether the reverse shooting approach is the best general way to find the stable trajectory of a model with saddlepath-type properties.

One of the big problems with the reverse shooting approach is that possible computational errors are introduced at a variety of different stages and these have the potential to compound causing the solution trajectory to "blow-up". This makes it an attractive proposition to compare these results with those derived using forward shooting, an alternative approach that attempts to find the solution by minimising all errors, including computational errors, simultaneously.

Our initial presumption was that the reverse shooting algorithm would always be more efficient than forward shooting because, for a given problem, the reverse shooting algorithm has to search over a smaller manifold. However, the results of this paper indicate that the problem of compounding errors is much more complicated than we had initially anticipated. We now think that it is possible that, in some cases, the forward shooting algorithm might be more efficient. Comparing outcomes under reverse shooting and forward shooting is the planned focus of our future research. In particular, the forward shooting approach would be expected to require more computational effort than reverse shooting, but may have a higher success rate.

REFERENCES

Anderson, G., and G. Moore (1985), "A Linear Algebraic Procedure for Solving Linear Perfect Foresight Models," *Economics Letters*, 17, pp. 247-252.

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

Boucekkine, R. (1995), "An Alternative Methodology for Solving Nonlinear Forward-Looking Models," *Journal of Economic Dynamics and Control*, 19, pp. 711-734.

Fair, R. C., and J. B. Taylor (1983), "Solution and Maximum Likelihood Estimation of Dynamic Nonlinear Rational Expectations Models," *Econometrica*, 51(4), pp. 1169-1185.

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

Judd, K. L. (1998), *Numerical Methods in Economics*, MIT Press, Cambridge, Massachusetts, USA.

Khalil, H. K. (1996), *Nonlinear Systems*, 2nd Edition, Prentice Hall, Upper Saddle River, U.S.A.

Lararias, J. C., J. A. Reeds, H. M. Wright, and P. E. Wright, 1998, "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions", *SIAM Journal of Optimization*, 9(1), pp.112-147.

MathWorks, 2002, website: www.mathworks.com,

Zadrozny, Peter A. (1998), "An Eigenvalue Method of Undetermined Coefficients for Solving Linear Rational Expectations Models," *Journal of Economic Dynamics and Control*, 22(8-9), pp. 1353-1373.

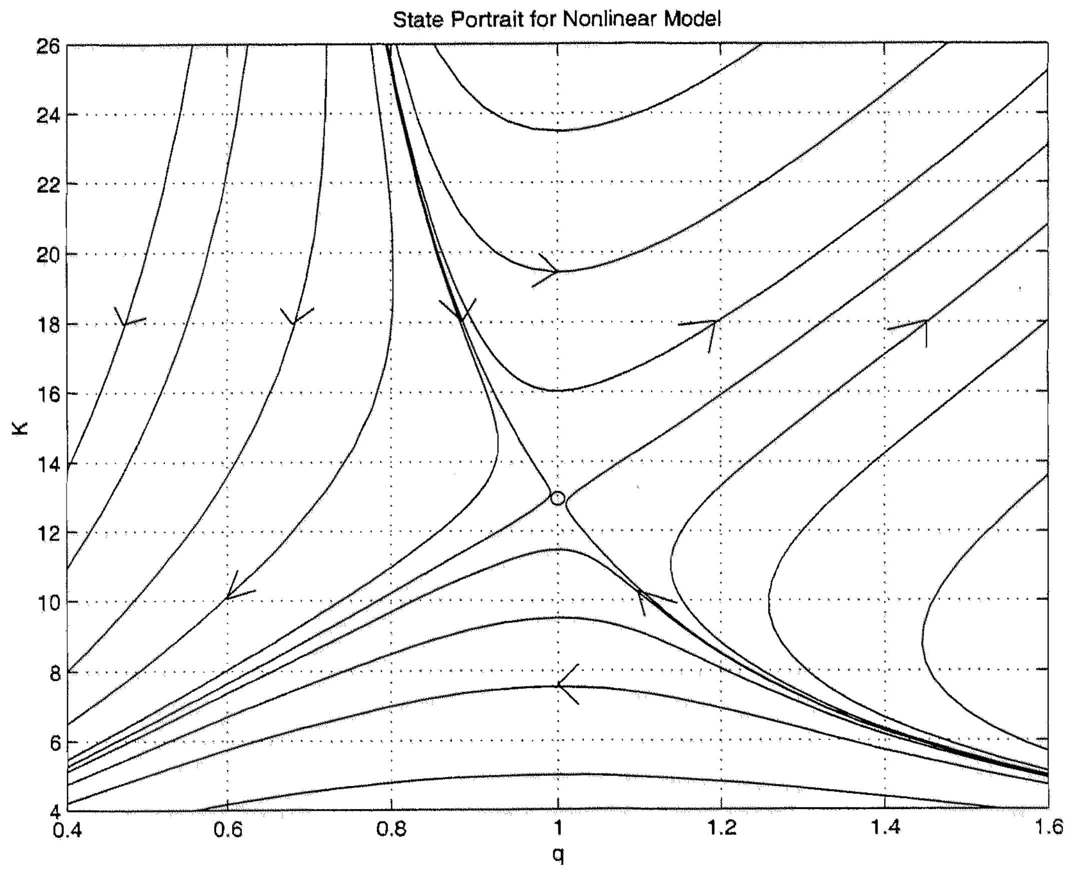


Figure 1. Phase Diagram of the Two-Dimensional Model
True (Non-Linear) Model

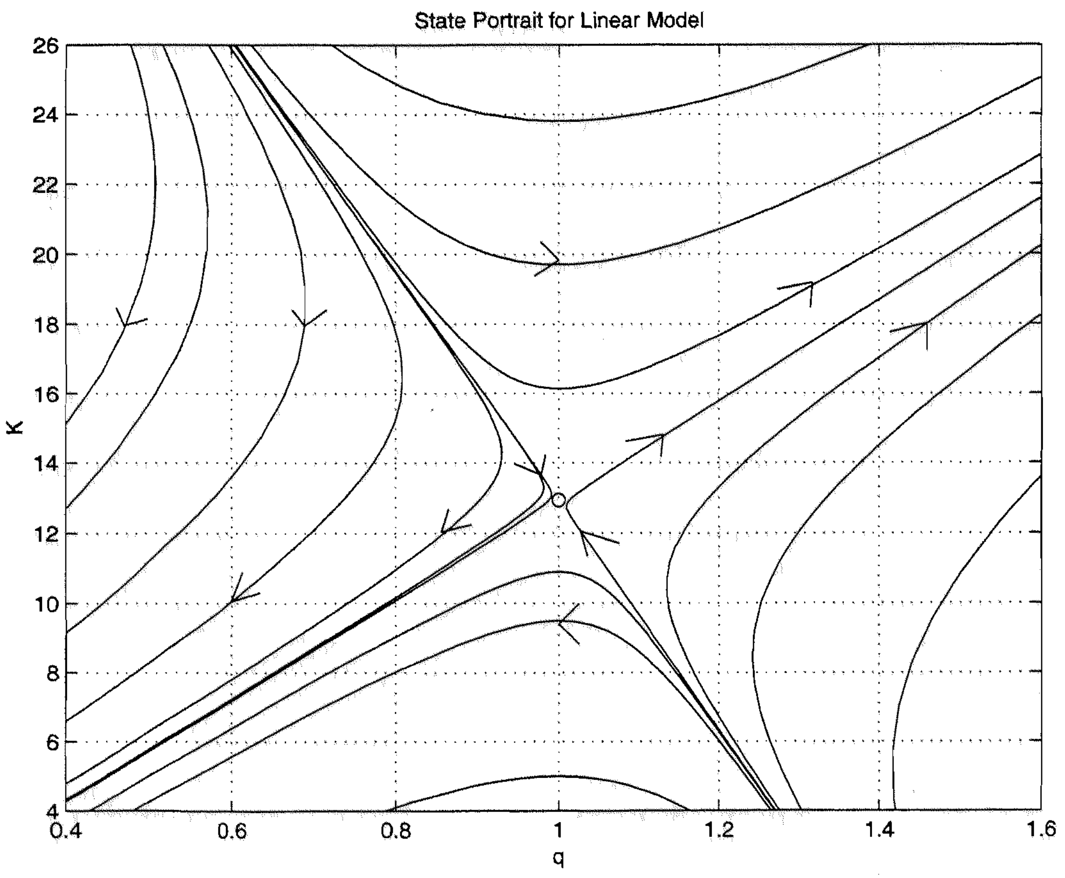


Figure 2. Phase Diagram of the Two-Dimensional Model
Linearised Model

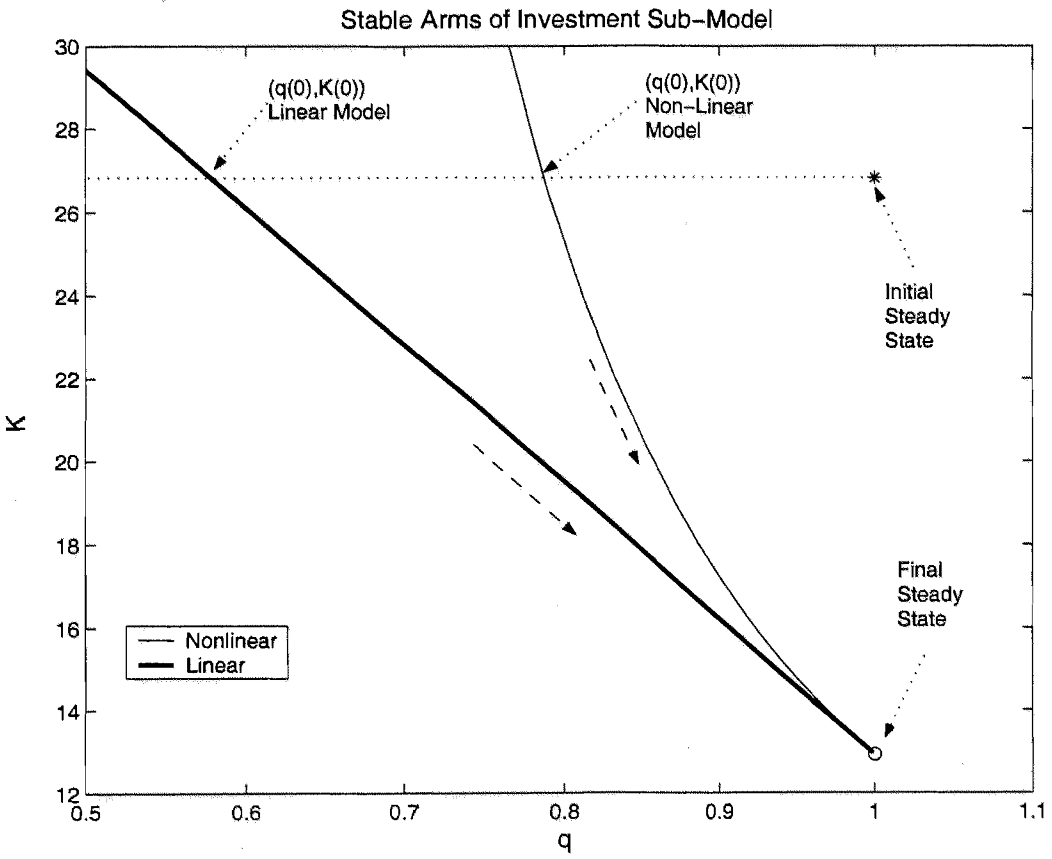


Figure 3. Stable Arms of the One-Dimensional Model.

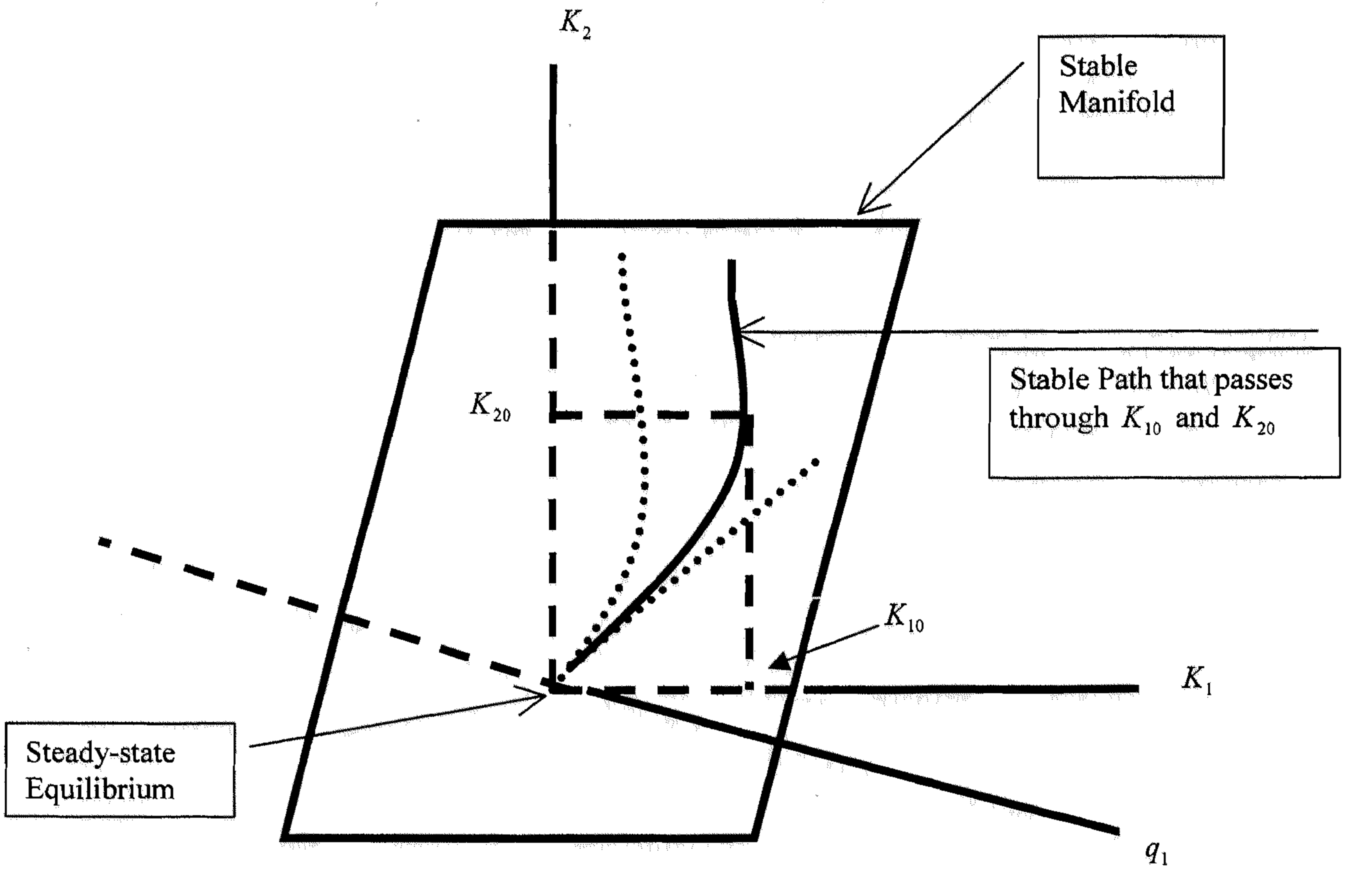


Figure 4. Visualising the Stable Manifold
The Case of Two Stable Eigenvalues and One Unstable Eigenvalue.

Search Space				
	Maximum Initial Capital Stock		Maximum Final Capital Stock	
Dim	Mean	S. Deviation	Mean	S. Deviation
10	2062.9237	553.6218	160.7431	36.3833
20	73.0469	15.9201	6.4759	0.9307
30	7.4386	1.1524	1.2953	0.0555
40	3.1230	0.3415	1.0000	0.0000

Table 1. Effects of Dimensionality on Search Space.

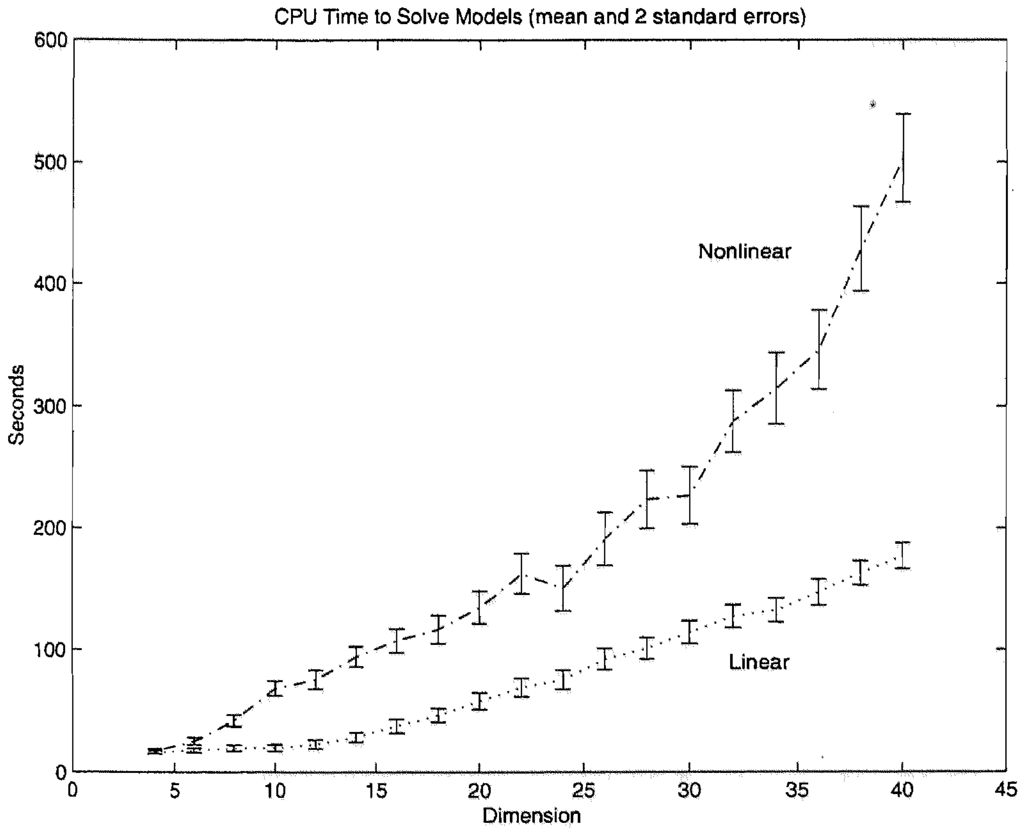


Figure 5. Computational Effort.
Results are for all models.

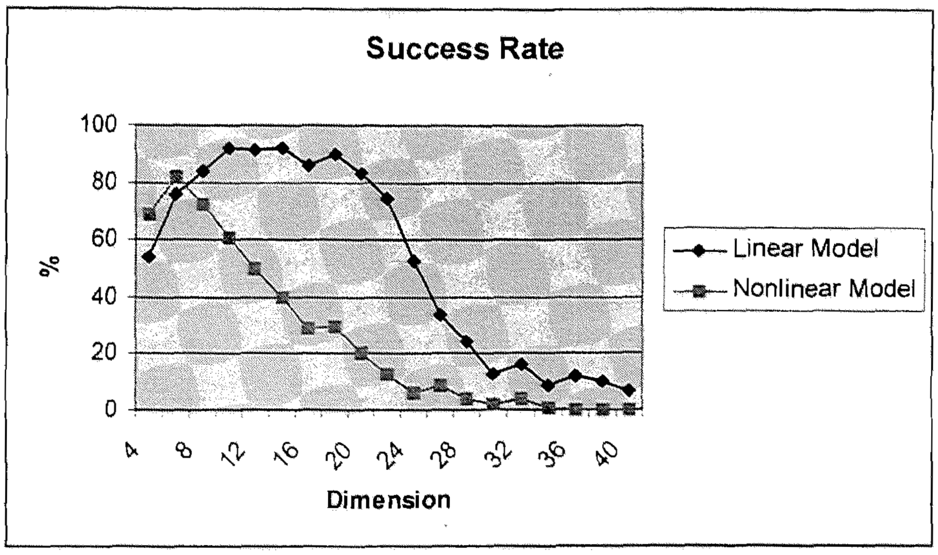
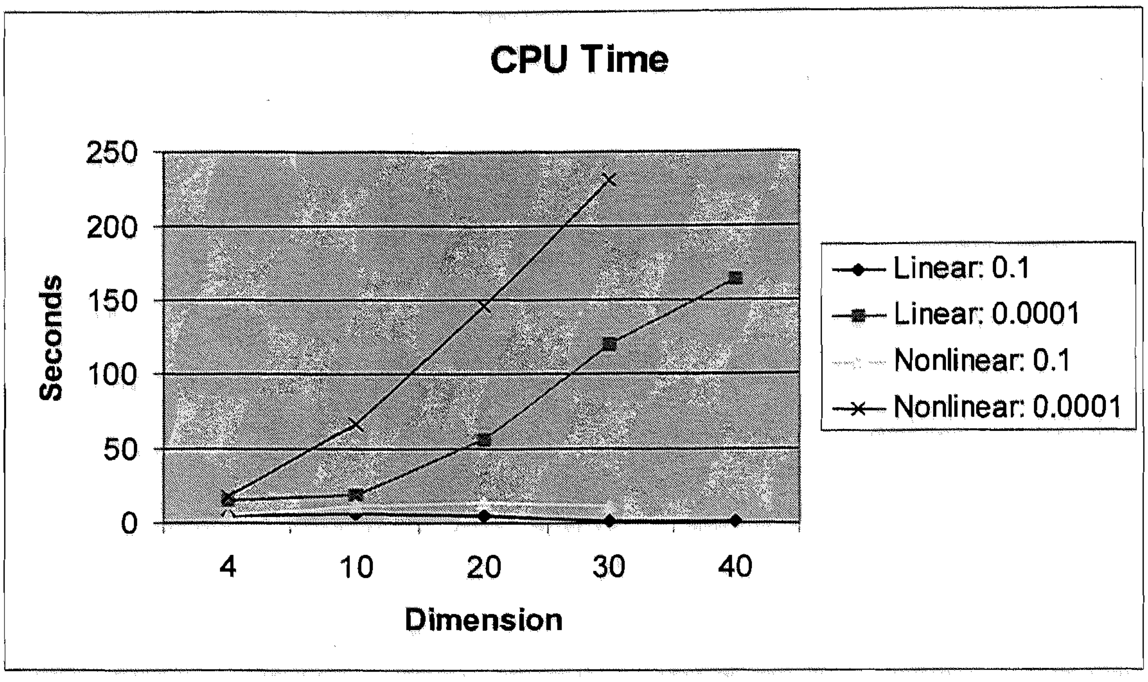


Figure 6:
Success is Defined as Successfully Solving Model (Successful Search + Successful Forward Trajectory); Search tolerance = 0.0001



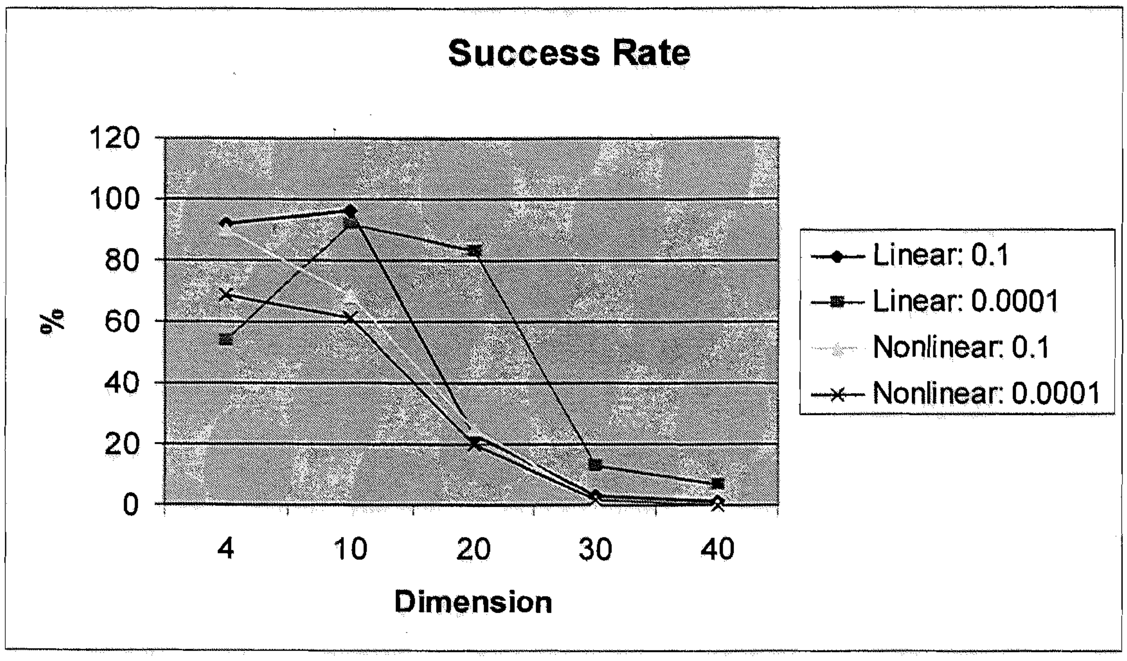


Figure 7:
Varying Search Tolerance: CPU Time and Success Rate.
Results are for Successfully Solving Model (Successful Search + Successful Forward Trajectory).

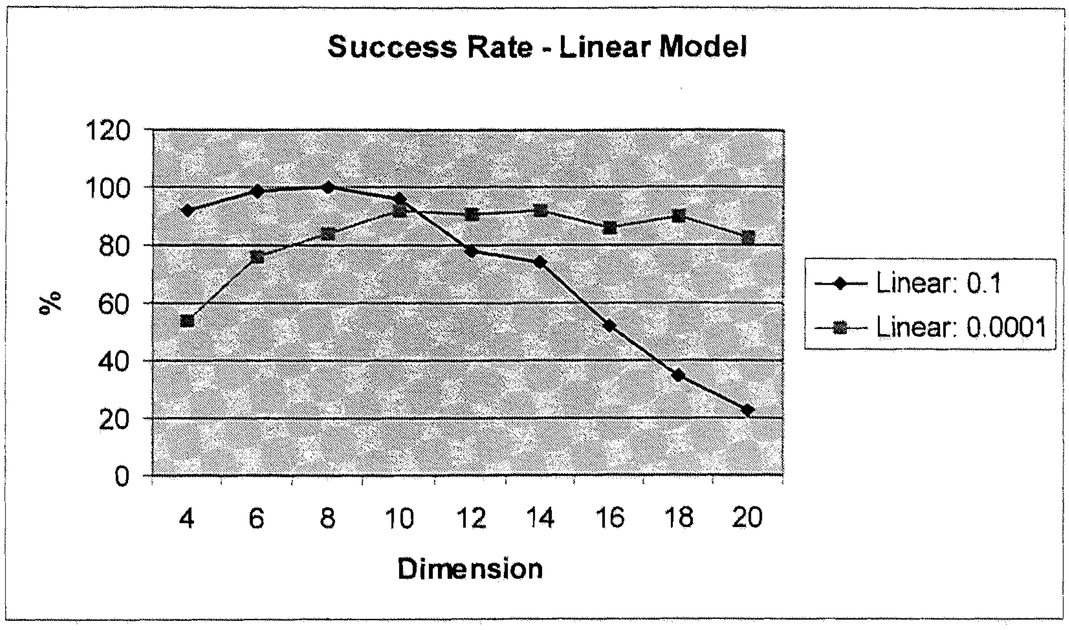


Figure 8:
Varying Search Tolerance: Success Rate for Linear Model at Lower Dimensions. Results are for Successfully Solving Model (Successful Search + Successful Forward Trajectory).

RESEARCH PAPER SERIES - RECENT PUBLICATIONS IN THE DEPARTMENT OF ECONOMICS

NO.	AUTHOR/S	TITLE	DATE	INTERNAT. WORKING PAPER NO.	ISBN NO.	TOTAL NO. OF PAGES
850	Bruce Phillips & William Griffiths	Female Earnings and Divorce Rates: Some Australian Evidence	July 2002	IWP 787	0 7340 2505 X	29
851	Robert Dixon	A Discussion of the Appropriate Method for Decomposing Changes Over Time in a Weighted Aggregate into its Proximate Determinants and an Application to Male Participation Rate Changes	July 2002	IWP 788	0 7340 2506 8	23
852	Jenny N. Lye & Joseph G. Hirschberg	Tests of Inference for Dummy Variables in Regressions with Logarithmic Transformed Dependent Variables	July 2002	IWP 789	0 7340 2507 6	16
853	Hsiao-chuan Chang	Are Foreign Workers Responsible for the Increasing Unemployment Rate in Taiwan?	August 2002	IWP 790	0 7340 2508 4	23
854	J. G. Hirschberg & D. J. Slottje	Bounding Estimates of Wage Discrimination	August 2002	IWP 791	0 7340 2509 2	24
855	Yuan K. Chou	The Australian Growth Experience (1960-2000): R&D-Based, Human Capital-Based, or Just Steady State Growth?	August 2002	IWP 792	0 7340 2510 6	27
856	William Griffiths	A Gibb's Sampler for the Parameters of a Truncated Multivariate Normal Distribution	August 2002	IWP 793	0 7340 2511 4	16
857	Olan T. Henry, Nilss Olekalns, & Kalvinder Shields	Non-linear Co-Movements in Output Growth: Evidence from the United States and Australia	September 2002	IWP 794	0 7340 2512 2	10
858	Ric D. Herbert & Peter J. Stemp	Reverse Shooting in a Multi-Dimensional Setting	September 2002	IWP 795	0 7340 2513 0	35