March 2009

A New Algorithm for Solving Dynamic Stochastic Macroeconomic Models*

Abstract

This paper introduces a new algorithm, the Recursive Upwind Gauss Seidel method, and applies it to solve a standard stochastic growth model in which the technology shocks exhibit heteroskedasticity. This method exploits the fact that the equations defining equilibrium can be viewed as a set of algebraic equations in the neighborhood of the steady-state. In a non-stochastic setting, the algorithm, in essence, continually extends a local solution to a globally accurate solution. When stochastic elements are introduced, it then uses a recursive scheme in order to determine the global solution. This method is compared to projection, perturbation, and linearization approaches and is shown to be fast and globally accurate. We also demonstrate that linearization methods perform poorly in an environment of heteroskedasticity even though the unconditional variance of technology shocks is relatively small and similar to that typically used in RBC analysis.

- *JEL Classification: C63,C68,E37*
- Keywords: *numerical methods, Gauss Seidel method, projection methods, real business cycles, crash state*

Victor Dorofeenko
Department of Economics and Finance
Institute for Advanced Studies
Stumpergasse 56,A-1060 Vienna, Austria

Gabriel S. Lee
IREBS
University of Regensburg
93053 Regensburg, Germany
And
Institute for Advanced Studies

Kevin D. Salyer
Department of Economics
University of California
Davis, CA 95616

*Corresponding Author: Kevin D. Salyer, Department of Economics, University of California, Davis, CA 95616. Phone: 530-752-8359, Fax: 530-752-9382, E-mail: kdsalyer@ucdavis.edu.

We thank the participants at the 2006 CEF conference, especially Kenneth Judd, and two anonymous referees for their comments. We gratefully acknowledge the financial support from Jubiläumsfonds der Oesterreichischen Nationalbank (Jubiläumsfondsprojekt Nr. 11042).

1 Introduction

Modern quantitative macroeconomics necessarily involves the use of numerical methods in order to compute the equilibrium behavior of a model economy. As initially introduced by Magill (1977) and later used by Kydland and Prescott (1982) in their seminal work on business cycle models, linearization methods have been the preferred solution approach. Such methods are easy to implement and, as shown by Christiano (1990), do not introduce significant approximation errors for many settings studied by macroeconomists.

But, as discussed in Judd (2002), linearization methods are not trouble free. Quoting from that paper: "For example, Tesar (1995) uses the Kydland-Prescott method and found an example where completing asset markets will make agents worse off. This result violates general equilibrium theory and can only be attributed to the numerical method used." (p.2) More recently, Kim and Kim (2003) have shown that this error in welfare analysis is symptomatic of linearized models and argue in favor of second-order approximation methods; variations on this theme have been developed by Sims (2000), Schmitt-Grohe and Uribe (2004) and Aruoba et al. (2006).

In order to study more complicated settings, non-linear methods have also been proposed that employ either projection techniques (Judd (1992), McGrattan (1999) and Christiano and Fisher (2000)) or perturbation techniques (Judd and Guu (1997)). In a recent contribution to understanding these approaches, Aruoba et al. (2006) examine the accuracy of these methods (along with traditional linearization and log-linearization methods) within the context of a prototypical real business cycle model. Their results replicate Christiano's earlier analysis (1990) in that, for economies characterized by low risk aversion (i.e. little curvature in

the utility function) and shocks that do not push the economy far from the steady-state (i.e. small variance of technology shocks), linearization methods do quite well.[1] However, linearization methods, not surprisingly, deteriorate quickly in the presence of large shocks and high risk aversion.

This paper complements and extends the analysis by Aruoba, et al. (2006) in two dimensions. The first and major contribution of this paper is the introduction of a new algorithm to solve stochastic dynamic economies; for our analysis we use a standard real business cycle model (again, with the shocks following a discrete-state Markov process). Our approach involves two parts: first, a one-pass continuous modification of the Upwind Gauss-Seidel (UGS) Algorithm (Judd (1998)) is used to solve for the non-stochastic problem, and, second, an implicit iterative scheme is employed to account for the stochastic effects. In the latter iterative approach, the small numerical magnitudes of the stochastic terms (e.g. cross-state transition probabilities in the case of discrete-state Markovian processes or variances for continuous AR processes) produces relatively fast convergence. We refer to this procedure as the RUGS (Recursive Upwind Gauss-Seidel) method. The algorithm has two strengths: (1) It is computationally fast; and (2) It has high global (i.e. non-local) accuracy. The essence of the RUGS approach, in contrast to other common nonlinear methods, is that it continually extends a local linear solution (in the neighborhood of the steady-state) to a globally accurate one over an arbitrary interval. This is an improvement over perturbation methods which aim to increase the accuracy of a local solution; at the same time, the RUGS method is more tractable than other globally accurate approximations such as projection methods.[2]

[1] Log linearization, however, performs the worst.

[2] We are grateful to an anonymous referee for providing this characterization of the RUGS approach.

We test the performance and accuracy of our algorithm in comparison with other popular nonlinear methods using the analysis by Aruoba et al.(2006) as a template.

In particular, we consider the following methods as comparison tests for our algorithm:[3]

1. A modification of the Value Function Iteration Algorithm (VFI) as it is implemented in Danthine et al. (1989), which is used here mostly to produce a standard time unit for other more advanced methods.

2. A perturbation method based on the Taylor expansion near the deterministic equilibrium point as described in Judd and Jin (2002).

3. A projection method using Chebyshev Polynomials spectral expansion of the sought policy functions. The projection of the residual is performed by the collocation procedure.

4. Standard linearization and log-linearization approximation methods.

The second contribution of the paper is that we examine discrete state settings so that heteroskedasticity in the technology shock can be introduced. In particular, we examine a crash state scenario and demonstrate that linearization methods perform poorly in this environment. We show that, even though the magnitude of the unconditional variance of the technology shock would lead one to conjecture reasonably small approximation errors (as suggested by Christiano (1999) and Aruoba et al. (2006)) for linear methods, the volatility of the conditional variances undermines this conjecture. Recent papers by Barro (2006) and

[3] We exclude from the benchmark set the Finite Element Method also considered in Aruoba et al., (2006) because, as shown there, in the case of smooth policy functions, it does not do better than the spectral expansion with Chebyshev Polynomials. Another interesting non-local method we don't test here is the Pade approximation considered in Judd and Guu (1997) for a simple deterministic capital growth problem.

Bloom (forthcoming) have argued forcefully for the presence of large shocks to uncertainty in the economy and, hence, our analysis motivates the use of more sophisticated solution methods in such settings.

Our results can be summarized follows. First, as in Aruoba et al. (2006), the local methods such as the linear approximation and the perturbation method perform poorly in the cases where non-local properties of the solution are essential. Compared to RUGS, the Perturbation method (5th order polynomial) is slower and less globally accurate (in terms of the Euler Equation Errors) even for simple two-state (i.e. homoskedastic) economies characterized by low risk aversion. Second, the RUGS method and the Projection method both produce globally precise solutions, but RUGS is faster in higher dimensional settings. In particular, as described below, an advantage of the RUGS method is that computational time increases linearly with the number of exogenous states.[4]

2 The Benchmark Problem

The benchmark problem for the algorithm is a discrete-state version of the familiar real business cycle model characterized by the following social planner problem:

$$\max_{c_t, n_t} E_0 \left[\sum_{t=0}^{\infty} \beta^t u\left(c_t, 1 - n_t\right) \right] \tag{1}$$

$$\text{subject to } c_t = \lambda_t f\left(k_t, n_t\right) + \left(1 - \Omega\right) k_t - k_{t+1}$$

[4] This becomes relevant when computing the implied distribution for capital in discrete state models. The histogram for this distribution will be bi-modal in a two-state model. If one wants to replicate a single-peaked distribution, then at least nine states (for the technology shock) were required. Solving this using a Projection method becomes quite time consuming.

where c_t, n_t, k_t, and λ_t denote individual consumption, labor hours, beginning-of-period capital stock and the technology shock, respectively; the functions $u(\cdot)$ and $f(\cdot)$ are the one-period utility and production functions; the constants β and Ω represent respectively agents' discount factor and the depreciation rate of capital. As in Danthine, Donaldson & Mehra (1989), the technology shock, λ_t, takes on the discrete set of values $\Lambda = (\lambda_1, \lambda_2, ..., \lambda_s)$ and follows a Markov process with the transition probability matrix P.

To apply our numerical algorithm, first rewrite eq. (1) as a Bellman Equation. The value function $V(k, \lambda)$ is defined by (with consumption eliminated via the (always binding) resource constraint):

$$V(k,\lambda) = \max_{k',n} \left[U\left(k, k', n, \lambda\right) + \beta \sum_{\lambda' \in \Lambda} P_{\lambda\lambda'} V\left(k', \lambda'\right) \right] \tag{2}$$

with

$$U\left(k, k', n, \lambda\right) = u\left(\lambda f(k, n) + (1 - \Omega) k - k', 1 - n\right). \tag{3}$$

The associated necessary conditions are (with U_i denoting the partial derivative with respect to the ith argument):

$$U_2\left(k, k', n, \lambda\right) + \beta \sum_{\lambda' \in \Lambda} P_{\lambda\lambda'} V_1\left(k', \lambda'\right) = 0 \tag{4}$$

$$U_3\left(k, k', n, \lambda\right) = 0 \tag{5}$$

$$V_1\left(k', \lambda\right) = U_1\left(k, k', n, \lambda\right) \tag{6}$$

Note that the set of necessary conditions form a complete differential-algebraic system of

equations for the sought value function, $V(k,\lambda)$, and policy functions $k' = k_\lambda(k)$ and $n = n_\lambda(k)$. Combining these we have:

$$U_2\left(k, k_\lambda(k), n_\lambda(k), \lambda\right) + \tag{7}$$

$$\beta \sum_{\lambda' \in \Lambda} P_{\lambda\lambda'} U_1\left[k_\lambda(k), k_{\lambda'}\left(k_\lambda(k)\right), n_{\lambda'}\left(k_\lambda(k)\right), \lambda'\right] = 0$$

$$U_3\left(k, k_\lambda(k), n_\lambda(k), \lambda\right) = 0 \tag{8}$$

3 The Algorithm

3.1 Solving the Deterministic Problem

Consider first the steady-state system of the economy in which $\lambda = 1$; then denoting $n_1(k) = n(k)$, eqs. (7) and (8) become:

$$U_2\left(k, k'(k), n(k)\right) + \beta U_1\left(k'(k), k'\left(k'(k)\right), n\left(k'(k)\right)\right) = 0 \tag{9}$$

$$U_3\left(k, k'(k), n(k)\right) = 0 \tag{10}$$

As is well known, the presence of the nested terms $k'\left(k'(k)\right), n\left(k'(k)\right)$ in the above equations implies that, in general, the solution involves functional methods. However, certain properties of the solution permit the treatment of eqs. (9) and (10) as algebraic so that its solution can be found by a standard algorithm (see sections 12.4 and 12.5 in the monograph of Judd (1998) for the description of the Upwind Gauss-Seidel (UGS) method used here and an example of its application to a continuous-state deterministic Bellman equation). This is demonstrated

below:

The solution of the system defined by eqs. (9) and (10) which maximizes the right-hand side of the Bellman equation has the unique stationary point $k'(k_s) = k_s$ that satisfies the equations

$$U_2(k_s, k_s, n_s) + \beta U_1(k_s, k_s, n_s) = 0 \tag{11}$$

$$U_3(k_s, k_s, n_s) = 0 \tag{12}$$

In equations (11) and (12) we introduce the corresponding stationary value of labor $n_s = n(k_s)$. The stationary point can be found by applying a standard non-linear equation solution method to the above equations.

Given this solution, the stability and uniqueness of the steady-state implies the inequalities:

$$\begin{aligned} \forall k \quad &> \quad k_s : k'(k) < k \\ \forall k \quad &< \quad k_s : k'(k) > k \end{aligned} \tag{13}$$

Assume that we already have the solution to the system defined by the eqs. (9) and (10) over the interval $[k_s, k_c]$. That is, over this interval, we know the functions $k'(k) = \tilde{k}(k); n(k) = \tilde{n}(k)$; and now we extend the interval to $[k_s, k_r]$ where $k_r > k_c$. The first inequality in eq.(13) implies that for some interval to the right of k_c, i.e. $k_c < k < k_c + \delta$, the value of the sought function lies to the left of k_c; therefore the nested functions $k'(k'(k)), n(k'(k))$ of such $k \in [k_c, k_c + \delta]$ may be calculated using the known functions

$\tilde{k}(k), \tilde{n}(k)$. Then, for this interval, the system of eqs. (9) and (10) takes the form:

$$U_2\left(k, k'(k), n(k)\right) + \beta U_1\left(k'(k), \tilde{k}\left(k'(k)\right), \tilde{n}\left(k'(k)\right)\right) = 0 \tag{14}$$

$$U_3\left(k, k'(k), n(k)\right) = 0$$

Note, critically, that this system of equations does not involve nested functions so it can be treated as an algebraic equation and solved by an appropriate standard numerical method.

Obtaining in this way the solution over the interval $[k_c, k_c + \delta]$, it is possible to extend the solution interval to the right; recursive repetition of the procedure can be done until the desired boundary, k_r, is reached. Thus choosing the right point of the initial interval infinitely close to the stationary point, we can then step by step extend the solution to the endpoint. Clearly, the second inequality in eq. (13) allows us to apply the same procedure to the left of k_s. This procedure generates the solution over the entire interval $[k_l, k_r]$. As stated earlier in the Introduction, the use of the UGS method to solve for the equilibrium policy functions has a nice intuitive characterization: In the neighborhood of the steady-state we can approximate the solution as a pair of linear functions. Then, as the neighborhood is extended over the entire interval, the initial linear functions are modified to produce a globally accurate set of policy functions. Summarizing the explanation above (with an addition of certain technical details), to obtain the UGS solution at the interval $[k_l, k_r] \ni k_s$, one should proceed as follows:

1. Find the steady-state point k_s from the numerical solution of the algebraic equations (11) and (12).

2. Define a small interval $[k_s - \delta_0, k_s + \delta_0]$ near the steady-state point and obtain an

approximate linear solution at that interval: $k'(k) = k_s + \left.\frac{dk'(k)}{dk}\right|_{k=k_s}(k-k_s), n(k) = n(k_s) + \left.\frac{dn(k)}{dk}\right|_{k=k_s}(k-k_s)$. The accuracy of that solution is $\epsilon_0 \lesssim \delta_0^2$, so the choice of, e.g., $\delta_0 = 10^{-6}$ yields $\epsilon_0 \lesssim 10^{-12}$, which is more than enough for our purposes.[5]

3. Solve numerically the algebraic equation (14) at the interval $[k_s + \delta_0, k_r]$. This step has two specific characteristics.

   (a) The numerical method of solution should proceed sequentially along the interval from left to right similar to a numerical method solving an initial value problem of an ordinary differential equation (ODE). So, the methods for solving an initial value problem of a differential-algebraic equation (DAE) may be of use. Here we use the IDA method implemented in the NDSolve routine of the *Mathematica* program (for those preferring Matlab, C or Fortran programming the corresponding routine of the SUNDIALS package (Hindmarsh et al. (2004)) may be used as well).

   (b) The solution obtained at each step of the IDA method should be recorded as an "external function" $\left(\tilde{k}(k), \tilde{n}(k)\right)$ for immediate use at the next step. Note that the IDA method, like any other adaptive scheme solving ODE or DAE, produces a solution at a discrete set of points with variable step given a fixed accuracy of calculations. The solution at any point in between is usually obtained from high-precision piecewise-polynomial interpolation. The routine performing the

[5] The width δ_0 of the initial interval of solution (see step 2 above) limits the initial step h_0 of the IDA solver (see item 3b) from above because of the requirement $k'(k_s + \delta_0 + h_0) \leq k_s + \delta_0$, which yields the approximate inequality $h_0 \leq \delta_0 \left(1 - \left.\frac{dk'}{dk}\right|_{k=k_s}\right)\left(\left.\frac{dk'}{dk}\right|_{k=k_s}\right)^{-1}$, so choosing a small value for δ_0 may slow down the very initial stage of calculations.

interpolation should be used as that "external function".[6]

4. Repeat step 3 for the interval $[k_l, k_s - \delta_0]$ proceeding from right to left. Combining the solutions obtained at the steps 2, 3 and 4 produces the solution to the problem.

This algorithm allows one to solve the deterministic growth problem using a fast one-pass algorithm. Note that the described procedure employs the proper ordering of values of the state variable, k, and starts from the absorbing state k_s; hence by the classification of Judd (1998), it can be treated as a continuous modification of the Upwind Gauss-Seidel Algorithm. Unlike the example presented in Judd (1998), where the UGS method is used to solve the Bellman equation for the value function, we apply it here to the necessary conditions to obtain the policy functions and thus avoid the time consuming maximization procedure.

3.2 Extending to Stochastic Settings

The procedure described above cannot be trivially generalized to the stochastic problem defined by eq. (2) because

1. The stochastic problem has multiple stationary points, depending on the current value of the technological factor.

2. The inequalities in eq.(13) may simultaneously have opposite signs for different values of λ.

[6] A fixed-step method using linear approximations to the policy functions as an initial guess at each step may substantially simplify the program if writing from scratch. Nevertheless, the trade-off between ease of implementation and desired effectiveness should be taken into account. We are grateful to an anonymous referee for this observation.

These properties significantly complicate and slow down the algorithm when extended in a straightforward manner to the general problem. Consequently, we will use a modification of a simple iterative scheme that employs the one-pass algorithm described above. As it is demonstrated below, this scheme converges quickly to the desired accuracy so the time of calculation does not grow considerably.

Since the conditional probabilities in any state i sum to unity, it is the case that

$$\beta U_1 \left[k_\lambda \left(k\right), k_\lambda \left(k_\lambda \left(k\right)\right), n_\lambda \left(k_\lambda \left(k\right)\right), \lambda\right] = \beta \sum_{\lambda' \in \Lambda} P_{\lambda\lambda'} U_1 \left[k_\lambda \left(k\right), k_\lambda \left(k_\lambda \left(k\right)\right), n_\lambda \left(k_\lambda \left(k\right)\right), \lambda\right]$$

Adding this term to both sides of eq. (7) permits the system of equations to be expressed as:

$$U_2 \left[k, k_\lambda \left(k\right), n_\lambda \left(k\right), \lambda\right] + \beta U_1 \left[k_\lambda \left(k\right), k_\lambda \left(k_\lambda \left(k\right)\right), n_\lambda \left(k_\lambda \left(k\right)\right), \lambda\right] =$$

$$\beta \sum_{\lambda' \in \Lambda} P_{\lambda\lambda'} \left(U_1 \left[k_\lambda \left(k\right), k_\lambda \left(k_\lambda \left(k\right)\right), n_\lambda \left(k_\lambda \left(k\right)\right), \lambda\right] - \right. \tag{15}$$

$$\left. U_1 \left[k_\lambda \left(k\right), k_{\lambda'} \left(k_\lambda \left(k\right)\right), n_{\lambda'} \left(k_\lambda \left(k\right)\right), \lambda'\right]\right)$$

$$U_3 \left[k, k_\lambda \left(k\right), n_\lambda \left(k\right), \lambda\right] = 0 \tag{16}$$

Note that the left hand side of eq. (15) is simply the deterministic case described in the section above. Hence, we can find the solution to the stochastic setting by an implicit iterative method which solves the two equations in eqs. (15) and (16). This solution is, of course, the solution to the original problem given in eq. (2). The iterative method can be

summarized as:

$$LHS\left(k_{\lambda}^{(m+1)}, n_{\lambda}^{(m+1)}, k_{\lambda,\lambda}^{(m+1)}, n_{\lambda,\lambda}^{(m+1)}, \lambda\right) = RHS\left(k_{\lambda}^{(m)}, n_{\lambda}^{(m)}, k_{\lambda,\lambda'}^{(m)}, n_{\lambda,\lambda'}^{(m)}, \lambda'\right) \tag{17}$$

where $k_{\lambda}^{(m)} = k_{\lambda}^{(m)}(k), k_{\lambda,\lambda'}^{(m)} = k_{\lambda'}^{(m)}\left(k_{\lambda}^{(m)}(k)\right)$ represent the mth iteration of the policy functions and LHS and RHS refer to eqs. (15) and (16). The left-hand side of system (17) coincides with equations (9) and (10) of the deterministic case. It differs from these equations by the presence of a non-zero "external function" at the right-hand side only and, consequently, can be solved using the one-pass method described there. Also, since the right-hand side of eq. (15) contains the sum of addends proportional to $P_{\lambda\lambda'}$ with zero diagonal terms, we expect fast convergence for this scheme since the non-diagonal elements are typically small given the high persistence often assumed for the technology shock.

Thus the complete scheme of the numeric solution of the stochastic growth problem consists of the following steps:

1. For each value of λ_i, solve the correspondent deterministic problem (defined by eqs. (9) and (10)) with the one-pass UGS method in order to obtain the policy functions $k_{\lambda_i}^{(0)} = k_{\lambda_i}^{(0)}(k), n_{\lambda_i}^{(0)} = n_{\lambda_i}^{(0)}(k)$. Use them as the initial guess in the iterative algorithm.

2. Repeat the iterations defined by eq. (17) until the desired accuracy is reached.

Each of these steps requires an application of the one-pass algorithm described above.

In addition to the expected speed of convergence, the other advantage of this scheme is that equations (17) are always solved separately, for each value of λ, so the time of computation grows only linearly with the number of discrete states considered. The cost of this

benefit is the necessity to use the iterative process; also, this iterative process may diverge under some circumstances and for some initial guesses and that may sometimes narrow the applicability region of the method. Another advantage of our approach, as discussed in the Appendix (Section 6.31), is that it is not necessary to discretize the capital stock explicitly. Again, this helps to reduce the curse of dimensionality.

3.3 Convergence and Existence Proof of RUGS

A rigorous proof of convergence of the iterative process (17) would, in general, require the use of linear operator theory in an appropriately normed functional space; unfortunately, we will not provide that here. Instead, we present some intuitive reasoning about the properties of convergence. A brief sketch of the derivation and the convergence criteria is presented below, while a more detailed explanation can be found in the Appendix.

First we need to linearize equation (17) near the exact solution

$$\begin{aligned} k_{\lambda}^{(m)}(k) &= k_{\lambda}(k) + \hat{k}_{\lambda}^{(m)}(k), \\ n_{\lambda}^{(m)}(k) &= n_{\lambda}(k) + \hat{n}_{\lambda}^{(m)}(k) \end{aligned} \tag{18}$$

assuming that the difference between the exact solution $k_{\lambda}(k)$ and the approximate mth step solution $k_{\lambda}^{(m)}(k)$ is small, $\left|\hat{k}_{\lambda}^{(m)}(k)\right| \ll 1$. And to analyze the convergence of the correspondent linearized equations, these equations can be written in the form of

$$A_{\lambda}(k)\,\hat{k}_{\lambda}^{(m)}(k) + B_{\lambda}(k)\,\hat{k}_{\lambda}^{(m)}(k_{\lambda}) = C_{\lambda}(k)\,\hat{k}_{\lambda}^{(m-1)}(k) + \sum_{\lambda'} D_{\lambda\lambda'}(k)\,\hat{k}_{\lambda'}^{(m-1)}(k_{\lambda}), \tag{19}$$

where $A_\lambda(k)$, $B_\lambda(k)$, $C_\lambda(k)$ and $D_{\lambda\lambda'}(k)$ are certain functions of k and λ (see the Appendix for the definitions). The convergence estimate of the iterative equation (19) is done in the Appendix and produces the following criteria:

$$\max_{k,\lambda}\left|\hat{k}^{(m)}(k,\lambda)\right| \leq \hat{\lambda}\frac{\beta k'_m M}{1-\beta k'_m\left(1+s\hat{\lambda}\right)}\max_{k,\lambda}\left|\hat{k}^{(m-1)}(k,\lambda)\right| \tag{20}$$

where

$$\hat{\lambda}=\max_{\lambda}\sum_{\lambda'}P_{\lambda\lambda'}\left|\lambda'-\lambda\right|, k'_m=\max_{k,\lambda}\left|\frac{dk_\lambda(k)}{dk}\right|,$$

and M and s are numerical constants of order 1. The convergence of the iterative process occurs if the numerical coefficient in the right-hand side of inequality (20) is smaller than 1. The small factor $\hat{\lambda}$, which is approximately equal to the conditional standard deviation of the productivity shock, causes the fast convergence of our algorithm provided the difference $1-\beta k'_m$ is not too small. The last requirement is satisfied if the stability condition $\left|\frac{dk_\lambda(k)}{dk}\right|<1$ holds in the solution region.

4 Comparison of Algorithms

In this section we compare the performance of the proposed algorithm (RUGS) with the performance of the four numerical methods (linearization, projection, perturbation, and value function iteration (VFI)) mentioned in the Introduction; for a description of these alternative methods, we refer the reader to Aruoba et al (2006). (However, for the VFI method we follow Danthine, Donaldson & Mehra (1989), which uses a simpler implementation.) Note here that we use Chebyshev polynomials up to the 9th order for the projection method and a 5th order

Taylor expansion for the perturbation method.

Each computational method is used to solve the basic problem given in eq. (1). To facilitate comparison between our results and that of Aruoba et al. (2006), for the most part we use the same parameter values as employed in their analysis. In particular, we assume that the first-order autocorrelation of the technology shock is 0.95, i.e. $Corr(\lambda_t, \lambda_{t-1}) = 0.95$. This is a degree of persistence commonly assumed in DSGE models. For the unconditional standard deviation of the technology shock, we examine two settings: a low and commonly used value of $\sigma_\varepsilon = 0.007$ and a high degree of volatility in which $\sigma_\varepsilon = 0.035$ thereby implying $\sigma_\lambda = 0.022$ and $\sigma_\lambda = 0.11$. We then examine two settings that differ in the modeling of the Markov process for the technology shock. First we examine a homoskedastic environment, i.e. one in which the conditional second moments of λ_t are constant. This homoskedastic assumption is used for the linear, log-linear and perturbation methods in which the process is assumed to be continuous. We use a two-state (i.e. discrete state) process when solving the model using RUGS, VFI and Projection methods. Also in a homoskedastic setting, we use the same $AR(1)$ process used by Aruoba et al. (2006) so we can compare more easily the RUGS and Projection methods. We then examine a five-state discrete state setting with a low probability crash state which, therefore, introduces heteroskedasticity into the shock process. Table 1 presents the unconditional/conditional standard deviations of λ for the two-state and five-state models.[7] As can be seen, this last setting introduces significant variation in the conditional standard deviation of λ_t. Consequently, the role of nonlinearities in the

[7] The unconditional standard deviations for the technology shock in the discrete state models are not exactly equal due to the calibration of the crash state scenario. Our goal was to explore the implications of heteroskedasticity and, consequently, parameterized the Markov process to highlight this feature. The difference in unconditional volatility is not large enough to influence equilibrium behavior.

policy rules should be highlighted in this setting.

For the two-state process, the transition probability matrix and possible realizations are given by:

$$P = \begin{pmatrix} 0.975 & 0.025 \\ 0.025 & 0.975 \end{pmatrix}, \Lambda = (0.978, 1.022) \tag{21}$$

For the five-state process, states 1 and 4 are considered "normal" low and high technology shock states while states 2 and 3 are average technology shock states ($\lambda_2 = \lambda_3 = 1 = E(\lambda_t)$). State 5 is a crash state in which the technology shock takes on a very low value. States 2 and 3 differ in the conditional probabilities of going to the crash state; specifically we assume that the probability of a crash state in State 3 is twice that of State 2. This assumption introduces, as seen in Table 1, heteroskedasticity in the shock process.

(Insert Table 1 about here)

The transition probability matrix and possible realizations are given by:

$$P = \begin{pmatrix} 1-2p_1 & p_1/2 & p_1/2 & p_1 & 0 \\ p_1 & 1-2p_1-p_2-\pi & p_2 & p_1 & \pi \\ p_1 & p_2 & 1-2p_1-p_2-2\pi & p_1 & 2\pi \\ p_1 & p_1/2 & p_1/2 & 1-2p_1 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 \end{pmatrix} \tag{22}$$

$$\Lambda = (1-\delta, 1, 1, 1+\delta, 1-\Delta)$$

$$p_1 = 0.017, p_2 = 0.2, \pi = 0.005, \delta = 0.027, \Delta = 0.35$$

The multistate (nine-state) modeling of the continuous shock setup (see Appendix 6.2) uses a Hermite-Gauss collocation in λ and produces formally the same discrete-state representation as given in equations (7) and (8). However, the coefficients P_{ij} can not be interpreted as transition probabilities; indeed, some of these take on negative values. Instead, the procedure produces a parametrized expectation model with a continuous shock setup.

The production function is assumed to be Cobb-Douglas:

$$f(k,n) = k^{\alpha} n^{1-\alpha} \tag{23}$$

Utility takes the functional form:

$$u(c,1-n) = \frac{\left(c^{\theta}(1-n)^{1-\theta}\right)^{1-\tau} - 1}{1-\tau} \tag{24}$$

For all simulations, we use the same parameter values as Aruoba et al. (2006):

Parameter	β	Ω	α	τ	θ	ρ	σ_{ε}
Value	0.9896	0.0196	0.4	$(2;8)$	0.357	0.95	$(0.007;0.035)$

These again are common values and produce steady-state values for the capital output ratio and time spent in work activity consistent with U.S. data. The models were solved under the assumption of low ($\tau = 2$) and high ($\tau = 8$) values of relative risk aversion[8]. The value $\tau = 8$ for the nine-state model is combined with a high technology shock variance

[8] It differs from Aruoba et al. (2006), where $\tau_{\max} = 50$

($\sigma_\varepsilon = 0.035$); this comprises the "extreme case" studied in Aruoba et al. (2006).

For each of the methods, we compare both accuracy and the speed of convergence. For accuracy, we follow Aruoba et al. (2006) and define accuracy in terms of the Euler equation residual introduced in Judd (1992) and Judd and Guu (1997) as

$$EE = 1 - \frac{(u_c)^{-1}\left(E_t\left[\left(1-\Omega+r_{t+1}\right)u_c\left(c_{t+1}, 1-n_{t+1}\right)\right]\right)}{c_t}, \tag{25}$$

where $u_c \equiv \frac{\partial u}{\partial c}$ denotes the correspondent partial derivative. The implication is that the approximation error is expressed as a percentage of steady-state consumption. Using formulas (3) and (24), the above Euler equation (EE) error is expressed in our notation as:

$$EE\left(k,\lambda\right) = 1 - \left[-\beta\frac{\sum_{\lambda'\in\Lambda} P_{\lambda\lambda'} U_1\left[k_\lambda\left(k\right), k_{\lambda'}\left(k_\lambda\left(k\right)\right), n_{\lambda'}\left(k_\lambda\left(k\right)\right), \lambda'\right]}{U_2\left(k, k_\lambda\left(k\right), n_\lambda\left(k\right), \lambda\right)}\right]^{\frac{1}{\theta(1-\tau)-1}} \tag{26}$$

We examine the Maximal and Average Euler Equation Error. Maximal EE error is defined by:

$$\max EE = \max_{k\in[k_l,k_r],\lambda\in\Lambda} EE\left(k,\lambda\right) \tag{27}$$

where the interval $[k_l, k_r]$ is equal to the solution interval. The average EE error is defined as average of absolute value of $EE\left(k,\lambda\right)$ over a time series sample generated using the tested policy functions. We use a sample length of 30000 to minimize the dependence on a specific realization.

4.1 Speed of convergence

We turn first to an analysis of the speed of the algorithms. (We do not report these for the linear and log-linear procedures since these are virtually instantaneous.) Table 2 presents the results (time is measured in seconds) for the remaining procedures. (We only report these for the case where relative risk aversion is equal to 2; the results for the high risk aversion economies were almost identical.) Note that in the five state economy, we compare only the projection and RUGS methods since the perturbation approach is not appropriate (it assumes the technology shock is homoskedastic) and the value function method is too time consuming.[9]

(Insert Table 2 about here)

Hence, we see that the RUGS approach is comparable in convergence time to the other globally accurate method (i.e. projection) due to the one-pass aspect of the procedure; moreover, the RUGS performance improves in higher dimensional settings. The measurements have been done using the computer AMD Turion MT-30 (1.6GHz) with 1Gb of RAM under Windows XP.

4.2 Error

The maximal and average Euler equation errors for each procedure are given in Tables 3 and 4. Note that the numbers given are the negative of the actual (logarithmic) values. These numbers represent the percentage cost in term of steady-state consumption due to the

[9] For the projection method, we use fifth, sixth and ninth order polynomials for the two-, five-, and nine-state models, respectively

approximation. A value of 2, for example, implies a mistake of $1 for every $100 spent while a value of 4 implies a $1 mistake for every $10,000 spent.

(Insert Tables 3 and 4 about here)

It is clear from the numbers reported in both Tables that the projection method and RUGS produce roughly the same level of accuracy and this is significantly more accurate than the other procedures. A further comparison of these two methods is presented in Figure 1 in which the tradeoff between computational time and accuracy are presented. (Here, the comparison is for the extreme case economy.) Again, the advantage of the RUGS approach when greater accuracy is desired is clear.

Figure 2 presents the Euler Equation errors over the range of the capital stock with the shaded area representing a range of $3\sigma_k$ (top panel of Figure 2) and over the range of technology shocks $1 \pm 3\sigma_\lambda$ (bottom panel of Figure 2). The RUGS and Projection curves correspond to the homoskedastic extreme case ($\tau = 8$ and $\sigma_\lambda = 0.035$). These graphs duplicate the message of Tables 3 and 4: the globally accurate methods are preferred in setting with high risk aversion and high volatility.

A final comparison is in terms of the policy functions generated by the solution methods. Here we focus on a comparison of the RUGS approach and the linear approximation method since the latter is commonly employed in the literature. The policy functions for labor and investment (as a function of capital) from the two- and five-state models are presented in Figures 3 and 4. In the two state economy, the policy functions are shown for the case when $\lambda_t = \lambda_2$ while in the five state economy, we plot the policy functions for both states 2 and 3 where $\lambda_2 = \lambda_3 = E(\lambda_t)$ .The roles that heteroskedasticity and risk aversion play in affecting

the accuracy of the linear approximation methods is evident in the graphs. Recall that in the two models (i.e. two-state and five-state), the unconditional standard deviations of the technology shock are roughly equal (see Table 1) and roughly equal to the relatively low value used in the literature. In Aruoba et al. (2006), the linear approximation method worked well in this setting when relative risk aversion was also in a reasonable range (i.e. below 10). As the graphs demonstrate, this is not the case when the shock exhibits heteroskedasticity. In the two state, homoskedastic setting, all methods produce similar policy functions (although the errors in the policy functions from the linear method as one moves away from the steady-state level of capital are evident). However, in the five state, heteroskedastic setting, the linear approximation produces fairly significant errors in regard to the labor policy function when combined with a risk aversion parameter of 8. Since heteroskedasticity appears to be important in understanding asset pricing movements, solution methods that are accurate in such settings may be useful.[10] In the final graph (Figure 5), we demonstrate that in the extreme case of relatively high risk aversion ($\tau = 8.0$) and very high volatility in the technology shock ($\sigma_\varepsilon = 0.035$), the linear approximation method produces significant errors with regard to the policy functions.

5 Some Final Remarks

The solution method presented here is fast and, consequently, useful in discrete and continuous state settings characterized by heteroskedastic or homoskedastic shocks to the economy.

[10] Note that the slope of the policy function for investment shifts from negative to positive when risk aversion goes from 2 to 8. This is a standard result (see Aruoba et al. (2006)) and reflects the relative strengths of the income and substitution effects of capital as determined by the curvature of agents' utility function.

While we have presented the solution method in the context of a simple RBC framework, in principle the RUGS method can be extended to models with more than one endogenous state variable (such as lagged consumption in a model with habit persistence). Below we discuss some possible difficulties in such an extension and present suggestions in how they can be avoided.

To apply the UGS method we need to find the deterministic trajectories starting from the equilibrium point of the model. A model with n state variables has an $(n-1)$-dimensional set of such trajectories. The complications that arise with the multidimensional RUGS are due to that multiplicity.

1. The number of trajectories we need to calculate at each step is proportional to N^{n-1}, where N is the number of grid trajectories along one dimension (if we have m continuous shocks, the factor of N^m should also be added). So the method is subject to the curse-of-dimensionality; (however, this problem is common to any of the global accuracy methods, e.g. projection methods).

2. To obtain a set of trajectories using the UGS method, we need to start in the vicinity of the steady-state point. If some of the stable eigenvalues are close to 1 and others are not (e.g., $\lambda_1 = 0.8$, $\lambda_2 = 0.2$) then one can show that, near the steady-state, all the trajectories approach closely the curve corresponding to the largest eigenvalue and it will be difficult to distinguish them given a finite accuracy of calculations. To avoid that issue we may (a) extend the distance between the starting points of trajectories and the steady-state point using more precision than implied by a linear local approximation or (b) increase locally the working precision of calculations.

3. The stochastic shocks shift the solution from a fixed discrete grid of trajectories, so to calculate the expectations we need to approximate the solution between the grid trajectories. A good choice for this approximation method may be a Fourier interpolation or a spline interpolation since these produce high precision results for a relatively low number of nodes.

Hence, while the RUGS approach can be extended in theory to a multidimensional state setting, it is not a trivial extension.

As the heteroskedastic environment that was studied in the paper suggests, we think it would be interesting to use this model in a crash state scenario as discussed recently in Barro (2006) and Salyer (2007). In particular, the latter paper demonstrated that Lucas's (1987) analysis of the welfare costs of business cycles severely understates the costs of fluctuations when a rare but catastrophic state in consumption growth is present. But this exercise is limited in that consumption growth was assumed (as in Lucas (1987)) to be exogenous. The question is whether a crash state in technology would correspond to a crash state in consumption; related to this is the question of whether the capital stock would be significantly different (say to precautionary saving) in such an economy relative to an economy with homoskedastic technology shocks. We leave the analysis of these questions to future research; but the solution method developed here would be appropriate for analysis in these settings.

6 Appendix

6.1 Convergence and Existence proof of RUGS

The detailed form of equation (19) is:

$$\left(\frac{dk_{\lambda}(k)}{dk}\right)^{-1}\hat{k}_{\lambda}^{(m)}(k)\left[\Delta(1,2;k,\lambda)+\beta\frac{dk_{\lambda}(k)}{dk}\sum_{\lambda'\in\Lambda}P_{\lambda\lambda'}\Delta\left(k_{\lambda},\lambda'\right)\right]-\beta\hat{k}_{\lambda}^{(m)}(k_{\lambda})\Delta(1,2;k_{\lambda},\lambda)= \tag{28}$$

$$\beta\left[\hat{k}_{\lambda}^{(m-1)}(k)\sum_{\lambda'\in\Lambda}P_{\lambda\lambda'}\Delta\left(k_{\lambda},\lambda'\right)+\sum_{\lambda'\in\Lambda}P_{\lambda\lambda'}\hat{k}_{\lambda'}^{(m-1)}(k_{\lambda})\Delta\left(1,2;k_{\lambda},\lambda'\right)\right],$$

where

$$\begin{aligned}\Delta(k,\lambda) &= \Delta(1,1;k,\lambda)+\frac{dk_{\lambda}(k)}{dk}\Delta(1,2;k,\lambda), \qquad (29)\\ \Delta(i,j;k,\lambda) &= \frac{U_{ij}(k,k_{\lambda},n_{\lambda},\lambda)U_{i3}(k,k_{\lambda},n_{\lambda},\lambda)-U_{i3}(k,k_{\lambda},n_{\lambda},\lambda)U_{i3}(k,k_{\lambda},n_{\lambda},\lambda)}{U_{i3}(k,k_{\lambda},n_{\lambda},\lambda)},\end{aligned}$$

where $U_{ij}(k,k_{\lambda},n_{\lambda},\lambda)$ denotes the second partial derivative of function $U(.)$ with respect to the ith and jth arguments. We have used in the derivation the following from (15) and (16) identities

$$\Delta(1,2;k,\lambda)+\frac{dk_{\lambda}(k)}{dk}\left[\Delta(2,2;k,\lambda)+\beta\left(\Delta(k,\lambda)+\sum_{\lambda'\in\Lambda}P_{\lambda\lambda'}\Delta\left(k_{\lambda},\lambda'\right)\right)\right]=0, \tag{30}$$

$$U_{13}(k,k_{\lambda},n_{\lambda},\lambda)+U_{23}(k,k_{\lambda},n_{\lambda},\lambda)\frac{dk_{\lambda}(k)}{dk}+U_{33}(k,k_{\lambda},n_{\lambda},\lambda)\frac{dn_{\lambda}(k)}{dk}=0$$

To produce an estimate of (20) let's note that in general (28) has the form:

$$a(k)y(k) = b\left(\hat{K}k\right) y\left(\hat{K}k\right) + g(k), \tag{31}$$

where $\hat{K}k \equiv k_\lambda(k)$. Iterating (31) N times with the help of the substitution

$$y\left(\hat{K}^n k\right) = \frac{b\left(\hat{K}^{n+1}k\right)}{a\left(\hat{K}^n k\right)} y\left(\hat{K}^{n+1}k\right) + \frac{1}{a\left(\hat{K}^n k\right)} g\left(\hat{K}^n k\right),$$

we arrive at the relation:

$$y(k) = y\left(\hat{K}^{N+1}k\right) \frac{b\left(\hat{K}^{N+1}k\right)}{a(k)} \prod_{m=1}^{N} \frac{b\left(\hat{K}^m k\right)}{a\left(\hat{K}^m k\right)} + \frac{1}{a(k)} \sum_{n=0}^{N} \left[g\left(\hat{K}^n k\right) \prod_{m=1}^{n} \frac{b\left(\hat{K}^m k\right)}{a\left(\hat{K}^m k\right)} \right] \tag{32}$$

Note that in our case $\lim_{N\to\infty} \hat{K}^N k = k_s$. Then if $|\frac{b(k_s)}{a(k_s)}| < 1$, the proceeding to the limit $N \to \infty$ in (32) yields:

$$y(k) = \frac{1}{a(k)} \sum_{n=0}^{\infty} \left[g\left(\hat{K}^n k\right) \prod_{m=1}^{n} \frac{b\left(\hat{K}^m k\right)}{a\left(\hat{K}^m k\right)} \right] \tag{33}$$

Relation (33) produces the estimate:

$$|y(k)| \leq \max_k \frac{1}{1 - \max\limits_{k' \in [k_s, k]} \left| \frac{b(k')}{a(k')} \right|} \frac{\max\limits_{k' \in [k_s, k]} |g(k')|}{|a(k)|}$$

Substituting actual values of the terms in the above inequality yields estimate (20):

$$\max_{k,\lambda}\left|\hat{k}_{\lambda}^{(m)}(k)\right| \leq \hat{\lambda}\frac{\beta k'_m M}{1-\beta k'_m\left(1+s\hat{\lambda}\right)}\max_{k,\lambda}\left|\hat{k}_{\lambda}^{(m-1)}(k)\right|, \tag{34}$$

where

$$\hat{\lambda}=\max_{\lambda}\sum_{\lambda'}P_{\lambda\lambda'}\left|\lambda'-\lambda\right|, k'_m=\max_{k,\lambda}\left|\frac{dk_{\lambda}\left(k\right)}{dk}\right|,$$
$$s=\max_{k,\lambda,\lambda'}\left|\frac{\Delta\left(k_{\lambda},\lambda'\right)-\Delta\left(k_{\lambda},\lambda\right)}{\Delta\left(1,2;k,\lambda\right)\left(\lambda'-\lambda\right)}\right|, M=\max_{k,\lambda}\frac{\Delta'+\Delta'\left(1,2\right)+\Delta\left(1,2\right)}{\left|\Delta\left(1,2;k,\lambda\right)\right|}, \tag{35}$$
$$\Delta'=\max_{k'\in[k_s,k],\lambda'}\left|\frac{\Delta\left(k',\lambda'\right)-\Delta\left(k',\lambda\right)}{\lambda'-\lambda}\right|, \Delta'\left(1,2\right)=\max_{k'\in[k_s,k],\lambda'}\left|\frac{\Delta\left(1,2;k',\lambda'\right)-\Delta\left(1,2;k',\lambda\right)}{\lambda'-\lambda}\right|,$$
$$\Delta\left(1,2\right)=\max_{k'\in[k_s,k],\lambda}\left|\Delta\left(1,2;k',\lambda\right)\right|$$

We assume that the values like $\frac{\Delta(k,\lambda')-\Delta(k,\lambda)}{\lambda'-\lambda}$, $\frac{\hat{k}^{(m-1)}(k,\lambda')-\hat{k}^{(m-1)}(k,\lambda)}{(\lambda'-\lambda)\hat{k}^{(m-1)}(k,\lambda)}$ are of order 1. However in our setup λ takes a discrete set of values and the derivatives $\frac{\partial k_{\lambda}(k)}{\partial\lambda}$, $\frac{\partial\hat{k}_{\lambda}(k)}{\partial\lambda}$ are not defined. Nevertheless, the direct numerical calculations in our example show that this statement is true. One can show that the denominator of M, $\Delta\left(1,2;k,\lambda\right)>0$ if the utility $u\left(c,l\right)$ is a logarithmic or exhibits decreasing returns to scale, so that the constant M is finite. More precisely, expressing the function $U\left(k,k',n,\lambda\right)$ through $u\left(c,1-n\right)$ with the help of formula (3), we obtain:

$$U_{33}\left(k,k_{\lambda},n_{\lambda},\lambda\right)\Delta\left(1,2;k,\lambda\right)=\lambda u_{c}\left(\lambda w w_{k}u_{c,c}-\left(1-\Omega+\lambda r\right)w_{n}u_{c,c}+w_{k}u_{c,n}\right)-$$

$$\left(1-\Omega+\lambda r\right)\left(u_{c,c}u_{n,n}-u_{c,n}^{2}\right)<0,$$

$$U_{33}\left(k,k_{\lambda},n_{\lambda},\lambda\right)=\lambda u_{c}w_{n}-\left(\lambda w\sqrt{-u_{c,c}}-\sqrt{-u_{n,n}}\right)^{2}-2\lambda w\left(\sqrt{u_{c,c}u_{n,n}}-u_{c,n}\right)<0,$$

where $r=f_{k}\left(k,n\right),w=f_{n}\left(k,n\right)$ denote the interest rate and wage respectively and w_{k}, w_{n} etc. are the corresponding partial derivatives. The above inequalities follows from the estimate of the sign of each term in these relations provided the utility function is logarithmic or decreasing returns to scale and the production function is constant returns to scale (e.g., $\lambda w w_{k}u_{c,c}<0$ because $w>0,w_{k}>0,u_{c,c}<0$ and so forth).

6.2 Continuous Expectation Approximation Using Hermite Collocation

The common approach to approximate the conditional expectation operator

$$Eg\left(z\right)\equiv\int\rho\left(z|z'\right)g\left(z'\right)dz'$$

is to replace the integral in the right-hand side by a finite sum

$$Eg\left(z\right)=\sum_{j=1}^{N}w_{j}\left(z\right)g\left(z_{j}\right),\tag{36}$$

where the nodes z_{j} and weights $w_{j}\left(z\right)$ can be defined using various procedures (see, e.g., Tauchen (1986,1991)). In our case of an $AR\left(1\right)$ process for $z=\lambda-1$ with a Gaussian

distribution of shocks ε

$$Eg\left(z\right)\equiv\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}e^{-\frac{\varepsilon^{2}}{2}}g\left(\rho z+\sigma\varepsilon\right)d\varepsilon \tag{37}$$

it is convenient to use the finite-term Hermite expansion:

$$g\left(z\right)\approx\sum_{n=0}^{N-1}g_{n}H_{n}\left(\frac{z}{L}\right), \tag{38}$$

In this case, the integral on the right-hand side of (37) can be easily computed using the Gauss transformation of Hermite polynomials (Bateman, Erdélyi (1953)):

$$\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}e^{-\frac{\varepsilon^{2}}{2}}H_{n}\left(\frac{\rho z+\sigma\varepsilon}{L}\right)d\varepsilon=\left(\sqrt{1-2\left(\frac{\sigma}{L}\right)^{2}}\right)^{n}H_{n}\left(\frac{\rho z}{L\sqrt{1-2\left(\frac{\sigma}{L}\right)^{2}}}\right),$$

so we obtain:

$$Eg\left(z\right)=\sum_{n=0}^{N-1}\left(\sqrt{1-2\left(\frac{\sigma}{L}\right)^{2}}\right)^{n}g_{n}H_{n}\left(\frac{\rho z}{L\sqrt{1-2\left(\frac{\sigma}{L}\right)^{2}}}\right) \tag{39}$$

The Hermite-Fourier coefficients can be approximated using the Hermite-Gauss quadrature formula:

$$g_{n}=\frac{1}{2^{n}n!\sqrt{\pi}}\int_{-\infty}^{\infty}e^{-x^{2}}g\left(Lx\right)H_{n}\left(x\right)dx\approx\frac{1}{2^{n}n!}\sum_{j=1}^{N}w_{j}g\left(Lx_{j}\right)H_{n}\left(x_{j}\right), \tag{40}$$

where the nodes x_j are zeros of the Nth-order Hermite polynomial, $H_N\left(x_j\right)=0$, and the

weights w_j are:

$$w_j = \frac{2^{N-1} N!}{N^2 H_{N-1}^2 (x_j)}$$

Substituting (40) into (38) and (39) we arrive at the N-point collocation in z space:

$$g(z) \approx \sum_{j=1}^{N} G_j(z)\, g(Lx_j), \tag{41a}$$

$$Eg(z) \approx \sum_{j=1}^{N} p_j(z)\, g(Lx_j), \tag{41b}$$

where the basis functions $G_j(z)$ and $p_j(z)$ are:

$$G_j(z) = \sum_{n=0}^{N-1} \frac{1}{2^n n!} H_n\left(\frac{z}{L}\right) H_n(x_j) \tag{42a}$$

$$p_j(z) = \sum_{n=0}^{N-1} \frac{1}{n!} \left(\frac{1}{2}\sqrt{1-2\left(\frac{\sigma}{L}\right)^2}\right)^n H_n\left(\frac{\rho z}{L\sqrt{1-2\left(\frac{\sigma}{L}\right)^2}}\right) H_n(x_j) \tag{42b}$$

Note that the functions $G_j(z)$ satisfy the equalities $G_j(Lx_i) = \begin{cases} 1, & i = j \\ 0 & i \neq j \end{cases}$, so relation $(41a)$ is indeed an interpolation formula.

We will use the above relations to approximate by finite sum the continuous form of the Euler equation (7)

$$U_2(k, k_\lambda(k), n_\lambda(k), \lambda) + \beta \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{\varepsilon^2}{2}} U_1\left[k_\lambda(k), k_{\lambda'(\varepsilon)}(k_\lambda(k)), n_{\lambda'(\varepsilon)}(k_\lambda(k)), \lambda'(\varepsilon)\right] d\varepsilon = 0, \tag{43}$$

$$\lambda'(\varepsilon) - 1 = \rho(\lambda - 1) + \sigma\varepsilon$$

Substituting there

$$\lambda = 1 + z, \lambda'(\varepsilon) = 1 + \rho z + \sigma\varepsilon,$$

$$g(\rho z + \sigma\varepsilon) = U_1\left[k_\lambda(k), k_{\lambda'(\varepsilon)}(k_\lambda(k)), n_{\lambda'(\varepsilon)}(k_\lambda(k)), \lambda'(\varepsilon)\right]$$

and using formula (42b) for the discrete set of values $z_i = \lambda_i - 1 = Lx_i,\ i = 1, ..., N$, we obtain equation (7) with the transition matrix

$$P_{\lambda_i\lambda_j} = p_j(Lx_i) = \frac{2^{N-1}N!}{N^2 H_{N-1}^2(x_j)} \sum_{n=0}^{N-1} \frac{1}{n!} \left(\frac{1}{2}\sqrt{1 - 2\left(\frac{\sigma}{L}\right)^2}\right)^n H_n\left(\frac{\rho x_i}{\sqrt{1 - 2\left(\frac{\sigma}{L}\right)^2}}\right) H_n(x_j) \tag{44}$$

The natural choice of scaling factor $L = \sqrt{2}\sigma\left(1 - \rho^2\right)^{-1/2}$ i.e., roughly equal to the standard deviation of z, simplifies (44) to:

$$P_{\lambda_i\lambda_j} = \frac{2^{N-1}N!}{N^2 H_{N-1}^2(x_j)} \sum_{n=0}^{N-1} \frac{1}{n!} \left(\frac{\rho}{2}\right)^n H_n(x_i) H_n(x_j) \tag{45}$$

6.3 Notes on Programming

All algorithms and procedures were programmed using *Mathematica 5.0*; all programs are available from the authors.

6.3.1 RUGS

As described earlier, the basic one-pass algorithm of RUGS method requires the numerical solution of an algebraic system, depending on the continuous parameter k. The corresponding numerical algorithm available in *Mathematica* 5 is the Newton algorithm used in the

IDA package of the **NDSolve** function. The IDA package was implemented initially at the Lawrence Livermore National Laboratory as a part of the free open source package SUNDIALS. More detailed description of the method can be found in Hindmarsh et al. (2004), Brenan, Campbell & Petzold (1996) and in the user documentation to the IDA library, Hindmarsh, Serban, and Collier (2006). As stated in the references above, the IDA package solves the initial value problem for the canonical system

$$F\left(t, y\left(t\right), y'\left(t\right)\right) = 0, \tag{46}$$

$$y\left(t_0\right) = y_0, \tag{47}$$

$$y'\left(t_0\right) = y_1 \tag{48}$$

(with the vector functions F and y and the independent scalar variable t) using an adaptive version of the Backward Differentiation Formula (BDF) algorithm. Below we present a brief description of the algorithm based on the references above. Equation (14) is a special case of (46) with $t = k$, $y = \begin{pmatrix} k' \\ \\ n \end{pmatrix}$ and the left-hand side of (14) as the function $F = F\left(t, y\left(t\right)\right)$, which does not depend on the derivative $y'\left(t\right)$. I.e., it represents in that case an algebraic system of equations depending on a scalar continuous parameter.

For our application it is important to note that the NDSolve function employs an adaptive scheme, i.e. the numeric step of the solution is variable and depends on the required accuracy. The solution itself is returned in the form of a piecewise-polynomial interpolating function (given by the **InterpolatingFunction** object in Mathematica). Note that these features imply it is not necessary to create a discrete grid for the capital stock.

Note also that for those who prefer Matlab or C programming, (along with the SUNDIALS package) they may approximate the solution for the output and for the "external function" $\left(\tilde{k}\left(k\right),\tilde{n}\left(k\right)\right)$ in equation (14) using a piecewise polynomial Hermite interpolation. We add a brief description of this interpolation below.

BDF Method The IDA implementation of BDF method solves initial value problem (46) at the interval $T_0 \leq t \leq T_1$ at discrete number of points $T_0 = t_0 < t_1 < \ldots < t_n < \ldots < t_N = T_1$ using adaptive steps $h_n = t_n - t_{n-1}$. The essence of the BDF method is the approximation of the derivative y' by the values of the function at several previous nodes:

$$y'_n = h_n^{-1} \sum_{i=0}^{q} \alpha_{n,i} y_{n-i}, \tag{49}$$

where the coefficients $\alpha_{n,i}$ depend on the previous step sizes $h_{n-q+1}, \ldots, h_n$ and on the order q (see Brenan, Campbell & Petzold (1996) for exact formulas). The substitution of (49) into (46) produces the algebraic equation:

$$F\left(t_n, y_n, h_n^{-1} \sum_{i=0}^{q} \alpha_{n,i} y_{n-i}\right) = 0, \tag{50}$$

which is solved w.r.t. y_n using the Newton method. To describe briefly the sequence of operations performed to make the $(n+1)$th step of the method, we follow the documentation of IDA package (see, Hindmarsh, Serban, and Collier, 2006):

1. Define the next integer order $q \in [1, 5]$. At each step the algorithm uses the maximal possible order (from 1 to 5), which provides a convergent solution. The maximal

feasible value of q provides the maximal step of solution (and maximal performance of the algorithm) given the accuracy required.

2. Define the next tentative step size h_{n+1} using the equation $\max\left[\varepsilon\left(h_{n+1}\right), \varepsilon_I\left(h_{n+1}\right)\right] = \varepsilon_0$, where $\varepsilon\left(h_{n+1}\right)$ is the estimate of error to the solution at the point $t_{n+1} = t_n + h_{n+1}$ and $\varepsilon_I\left(h_{n+1}\right)$ is the maximal error of the approximation to the solution at the interval $t_n \le t \le t_{n+1}$ using q-point polynomial interpolation.

3. Find the first guess $y_{n+1}^{(0)} = y^{(0)}\left(t_{n+1}\right)$ to the solution at the next point t_{n+1} (predictor) using qth order polynomial extrapolation.

4. Find the corrected solution y_{n+1} (corrector) solving equation (50) by the Newton method [Judd's monograph] with the predictor $y_{n+1}^{(0)}$ as an initial guess.

We omit the details of the stages above referring to the documentation, Hindmarsh, Serban, and Collier (2006), because (a) these details do not clarify the RUGS algorithm (which can be implemented as well using another similar package) and (b) they are unnecessary for a user of the package. Note also that at the initial step $t = t_1$ the value of q is set to 1. Then it is increased up to 5 or smaller maximal feasible value when the number of precedent steps and convergence criteria allow that.

Note that the solution is approximated between the grid points $t_0, \ldots, t_N$ using a q-point Hermite polynomial interpolation, i.e. the interpolation involving the grid values of the sought function $y\left(t_{n-q+1}\right), \ldots, y\left(t_n\right)$ and its derivatives $y'\left(t_{n-q+1}\right), \ldots, y'\left(t_n\right)$. This approximation is chosen, because (a) it is computationally the simplest way to reach the required accuracy and (b) the calculation of the predictor $y_{n+1}^{(0)}$ (see step 3 of BDF routine

above) uses the same polynomial formula for extrapolation.

Note also that an adaptive scheme of step choice greatly increases the ratio "accuracy/computer time" and is a conventional workhorse of DAE/ODE solvers, however a more simple, fixed step algorithm may be implemented as well.

6.3.2 Projection

The numeric solution of nonlinear equations for projection coefficients is performed using the **FindRoot** function in *Mathematica*; this uses the Newton algorithm.

6.3.3 Perturbation

The method requires the sequential calculation of partial derivatives so that a programming language with the capability of symbolic calculations is highly useful. Hence the usage of *Mathematica* is critical for this method.

6.3.4 Value Function Iteration

The Value Function Iteration algorithm does not require any unique properties of *Mathematica*.

References

Aruoba, S.B., Fernández-Villaverde, J. and Rubio-Ramírez, J., 2006. Comparing Solution Methods for Dynamic Equilibrium Economies. Journal of Economic Dynamics and Control 30, 2477–2508.

Barro, R. J., 2006. Rare Disasters and Asset Markets in the Twentieth Century. Quarterly Journal of Economics 121, 823-866.

Bateman, H., Erdélyi, A., 1953. Higher Transcendental Functions. NY, Mc Grow-Hill.

Bloom, N. forthcoming. The Impact of Uncertainty Shocks. Econometrica.

Brenan, K. E., Campbell, S. L., and Petzold, L. R., 1996. Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. SIAM, Philadelphia, Pa.

Boyd, J.P.,2001. Chebyshev and Fourier Spectral Methods (Second Revised Edition), Dover Publications, Inc., Mineola, New York, USA.

Christiano, L.J.,1990. Linear-Quadratic Approximation and Value-Function Iteration: A Comparison. *J*ournal of Business Economics and Statistics 8, 99-113.

Christiano, L. J. and Fisher, J. D. M., 2000. Algorithms for solving dynamic models with occasionally binding constraints. Journal of Economic Dynamics and Control, 24, 1179-1232.

Danthine, J.-P., Donaldson, J. B., Mehra, R.,1989. On Some Computational Aspects of Equilibrium Business Cycle Theory. Journal of Economic Dynamics and Control 13, 449-470.

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R.,2004. Shumaker, and Woodward C. S., SUNDIALS: Suite of Nonlinear and Differential D. E. /Algebraic Equation Solvers, LLNL technical report UCRL-JP-200037.

Hindmarsh, A.C., Serban, R., and Collier, A., 2006. User Documentation for IDA v2.5.0, LLNL

Judd, K.L.,1992. Projection Methods for Solving Aggregate Growth Models. Journal of Economic Theory 58, 410-452.

Judd, K. L., and Guu, S-M., 1997. Asymptotic Methods for Aggregate Growth Models". Journal of Economic Dynamics and Control 21, 1025-1042.

Judd, K.L.,1998. Numerical Methods in Economics. MIT Press.

Judd, K. L., 1996. Approximation, Perturbation, and Projection Methods in Economic Analysis. in: Hans Amman, David Kendrick, and John Rust (Eds.), Handbook of Computational Economics, Vol. 1, North Holland, Amsterdam, pp 509-585.

Judd, K.L. & Jin, H.-H., 2002. Perturbation Methods for General Dynamic Stochastic Models. Mimeo, Hoover Institution.

Kim, J. and Kim, S., 2003. Spurious Welfare Reversals in International Business Cycle Models. Journal of International Economics 60, 471-500.

Kydland, F.E. and Prescott, E.C., 1982. Time to Build and Aggregate Fluctuations. Econometrica 50, 1345-1370.

Lucas, R. E., Jr., 1987, Models of Business Cycles. New York: Basil Blackwell.

Magill, J.P.M., 1977. A Local Analysis of N-Sector Capital under Uncertainty. Journal of Economic Theory 15, 211-219.

McGrattan, E.R., 1999. Application of Weighted Residuals Methods to Dynamic Economic Models. in: R. Marimon and A. Scott (eds), Computational Methods for the Study of Dynamic Economies, Oxford University Press.

Salyer, K.D., 2007. Crash States and Macroeconomic Priorities. Economics Letters 94, 64-70.

Schmitt-Grohe, S. and Uribe, M., 2004. Solving Dynamic General Equilibrium Models Using a Second-Order Approximation to the Policy Function. Journal of Economic Dynamics and Control 28, 755-775.

Sims, C.A., 2000. Second Order Accurate Solution of Discrete Time Dynamic Equilibrium Models. Mimeo. Princeton University.

Tauchen, G., 1986. Finite state Markov-chain approximations to univariate and vector autoregressions. Economics Letters 20, 177-181.

Tauchen, G., 1991.Quadrature-Based Methods for Obtaining Approximate Solutions to Nonlinear Asset Pricing Models. Econometrica 59, 371-396.

Tesar, L., 1995. Evaluating the Gains from International Risksharing. Carnegie-Rochester Conference Series on Public Policy 42, 95-143.

Table 1: standard unconditional/conditional standard deviations of λ

Number of states (N)	$s=2$		$s=5$	
Unconditional s.d.(σ_λ)	0.022		0.028	
Conditional s.d.	i	$\sigma_{\lambda,i}$	i	$\sigma_{\lambda,i}$
	1	0.022	1	0.0078
	2	0.022	2	0.025
			3	0.035
			4	0.0078
			5	0

Table 2: Speed of Convergence

Method	Time ($s = 2$)	Time ($s = 5$)	Time ($s = 9$)
Perturbation (5th order)	5	*na*	5
Value Function Iteration	77	*na*	*na*
Projection	1.9	16	528
RUGS	1.7	49	293

Table 3: Euler Equation Errors in the Low Risk Aversion Economy

Risk aversion ($\tau = 2$)	$s = 2$		$s = 5$		$s = 9$	
	Max EE	Average EE	Max EE	Average EE	Max EE	Average EE
Linear	3.0	4.7	*na*	*na*	*na*	*na*
Log-linear	2.5	3.7	*na*	*na*	*na*	*na*
Perturbation	4.8	4.9	*na*	*na*	*na*	*na*
Value Function Iteration	2.2	3.2	*na*	*na*	*na*	*na*
Projection	6.8	7.9	6.7	8.7	6.7	7.9
RUGS	6.8	7.2	6.3	7.0	6.4	6.6

Table 4: Euler Equation Errors in the High Risk Aversion Economy

Risk aversion ($\tau = 8$)	$s = 2$		$s = 5$		$s = 9$	
	Max EE	Average EE	Max EE	Average EE	Max EE	Average EE
Linear	1.3	3.1	*na*	*na*	*na*	*na*
Log-linear	2.0	3.3	*na*	*na*	*na*	*na*
Perturbation	1.9	4.3	*na*	*na*	*na*	*na*
Value Function Iteration	2.2	3.1	*na*	*na*	*na*	*na*
Projection	6.9	7.7	6.7	8.6	4.3	5.7
RUGS	6.6	6.9	6.2	6.7	5.0	6.0

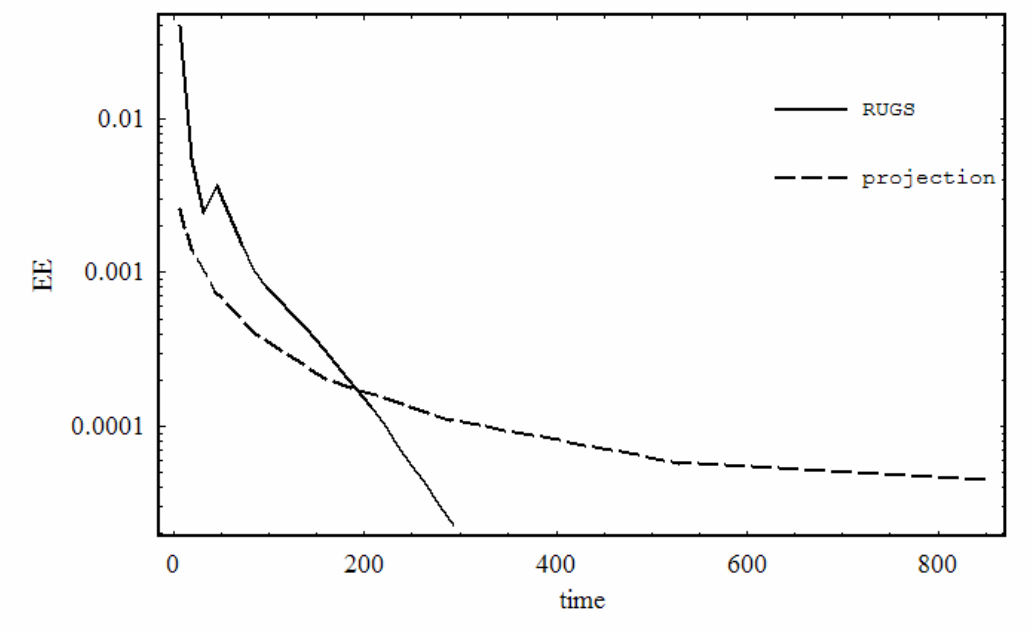

Figure 1: Maximal Euler Equation Error vs real computation time for RUGS and projection methods solving nine-state model, τ=8, σ=0.035

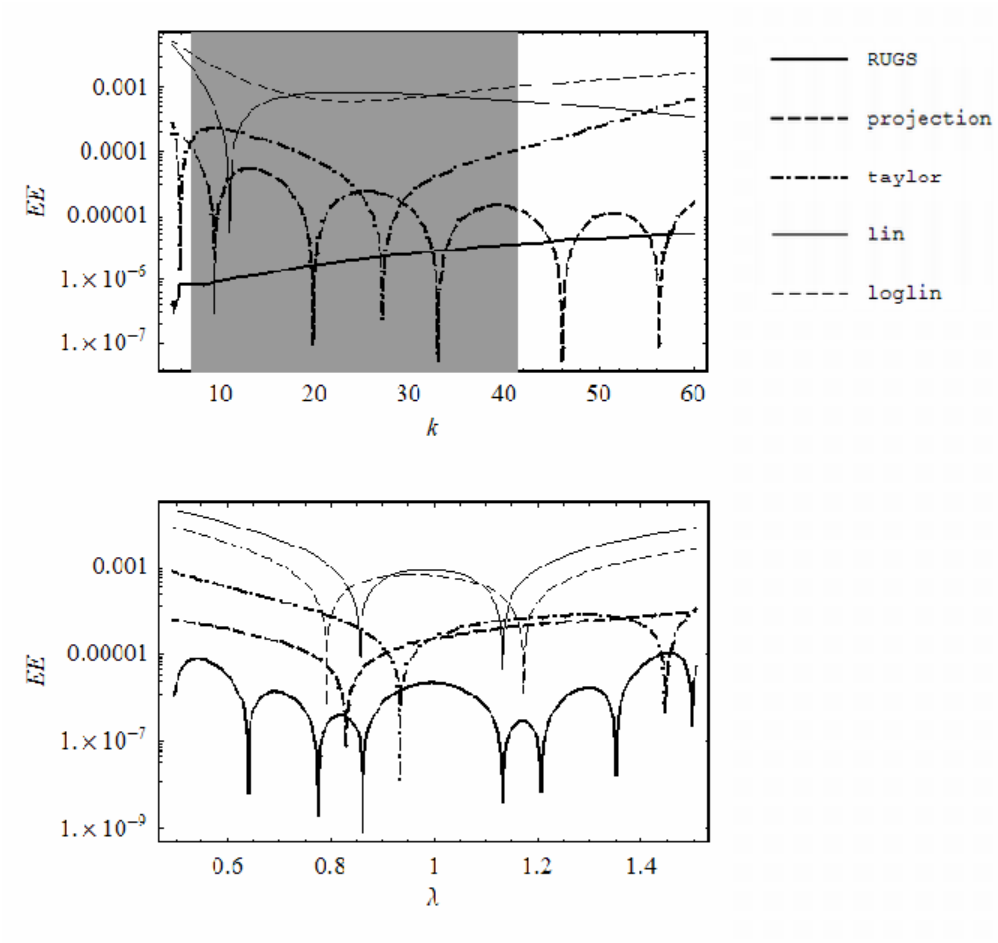

Figure 2: Euler Equation Error for different methods of solution: EE(k) for λ=1 and EE(λ) for k=k_s. τ=8, σ=0.035.

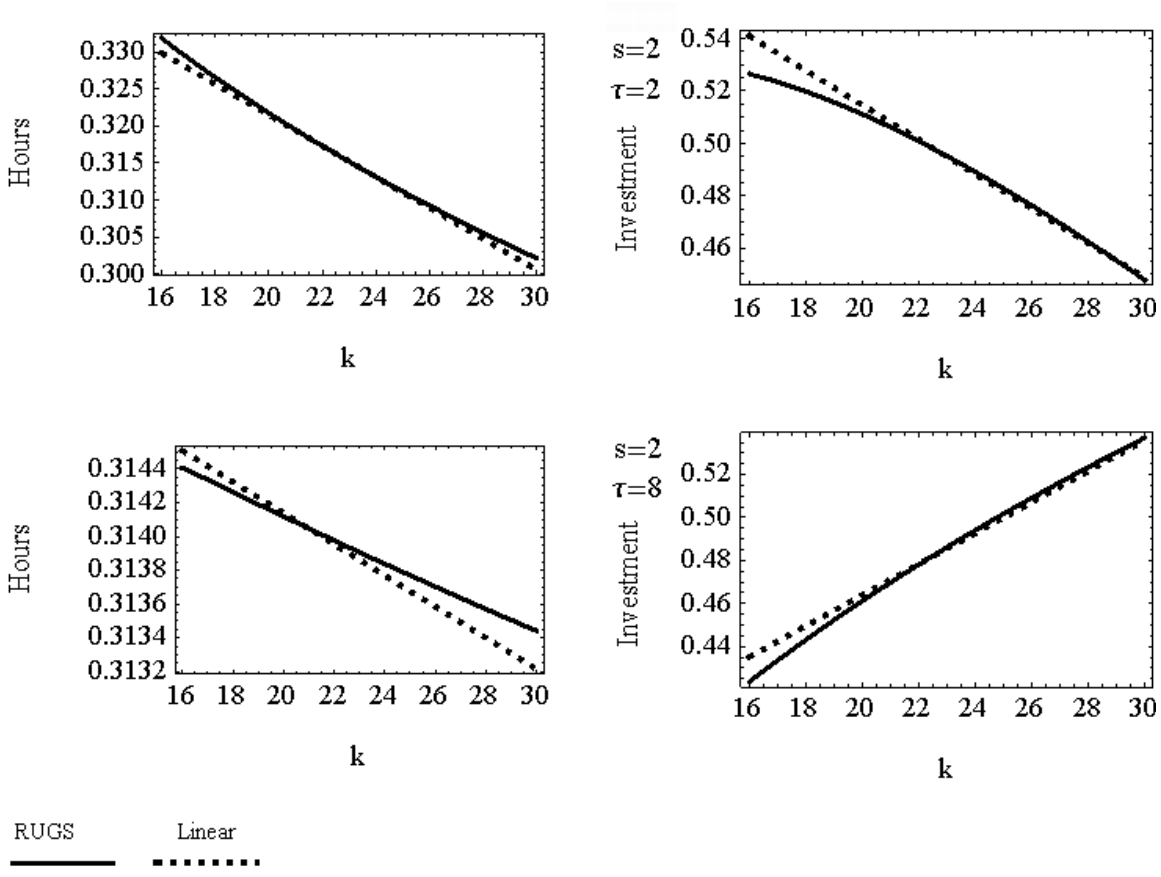

Figure 3: Policy functions (hours and investment) for the two-state model; $\lambda=\lambda_2$.

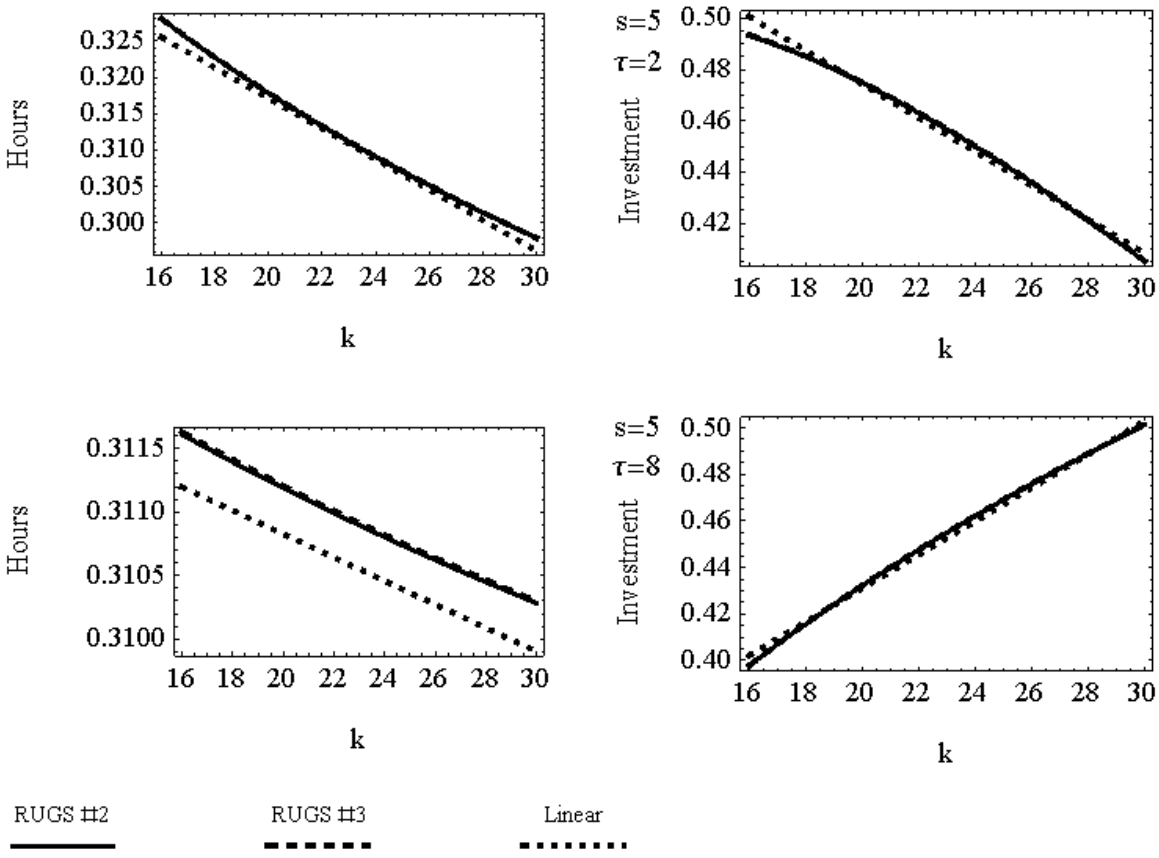

Figure 4: Policy functions (hours and investment) for the five-state model; $\lambda=\lambda_2$ (solid curve) and $\lambda=\lambda_3$ (dashed curve).

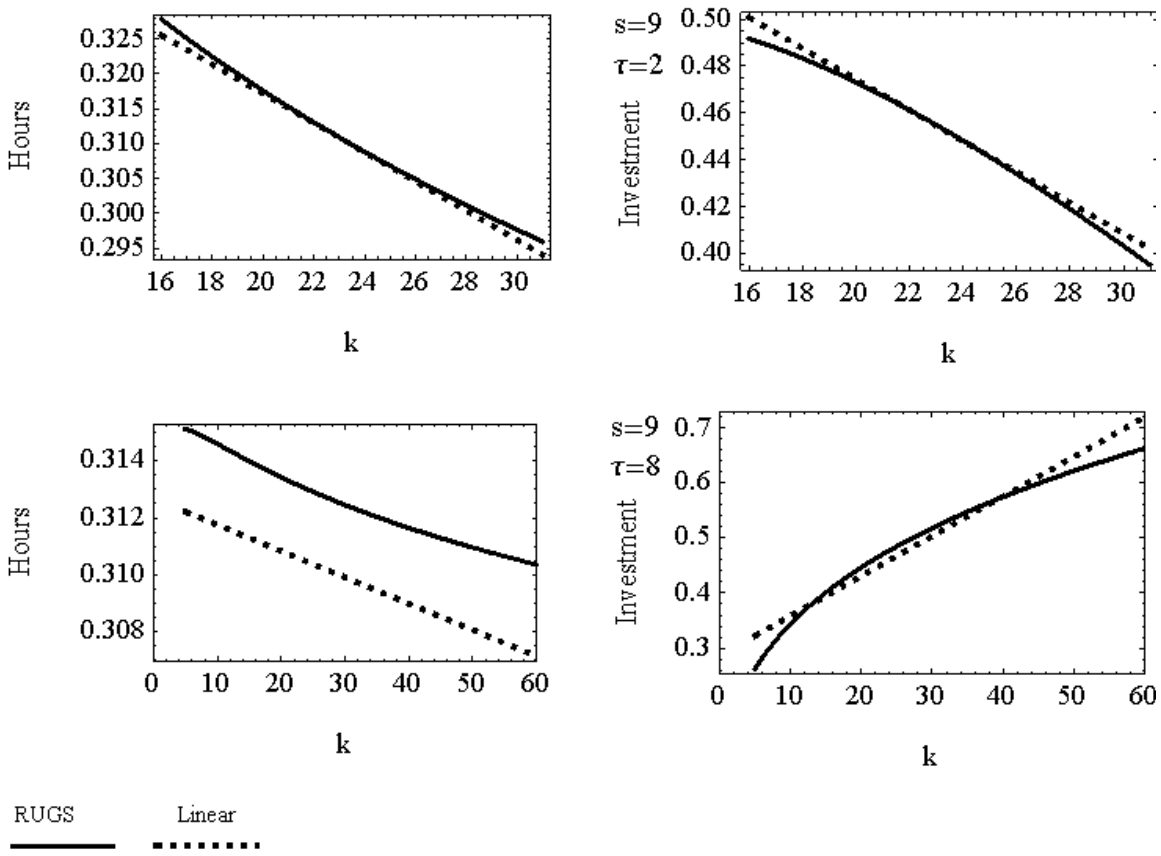

Figure 5: Policy functions (hours and investment) for the nine-state model, λ=1. The first row corresponds to τ=2, σ=0.007 and the second row corresponds to τ=8, σ=0.035.