Solving RBC Models by Second Order Approximation to the Policy Function

Master Thesis

In Pursuit of Master of Arts in Economics

Department of Economics
Humboldt University of Berlin

By

Kang, Long

MEMS

(Matrikel-Nr. 168292)

Examiners:

Prof. Harald Uhlig, Ph.D.

Mirko Wiederholt, Ph.D.

Berlin, August 15, 2003

Declaration of Authorship

I hereby confirm that I have authored this master thesis independently and without use of others than the indicated resources. All passages, which are literally or in general matter taken out of publications or other resources, are marked as such.

Kang, Long
Berlin, August 15, 2003

Solving RBC Models by Second Order Approximation to the Policy Function

Kang, Long
MEMS
Humboldt University of Berlin.

Abstract

This paper attempts to solve a benchmark real business cycle model by second order approximation to the policy function. After a brief summary of recent development in second order approximation in solving dynamic stochastic general equilibrium models, we choose Hansen's real business cycle model as a standard model and follow the approach proposed by Schmitt-Grohe & Uribe (2002) to solve for the recursive law of motion at second order. Then we do the impulse response and simulation experiment with the second order recursive law of motion and find that the impulse response at second order converges to a new level and the difference between first order and second order is very small in the standard calibration but becomes larger for other values of relative risk aversion. The calculated second moments tend to be very close between first order and second order for all parameters tested. Moreover, we conduct a brief theoretical comparison of the approach of approximation to the policy function with the usual approach at first order.

Contents

1. Introduction

It has been well established to solve dynamic stochastic general equilibrium (DSGE) models by first order approximation. And first order approximation is widely applied, as it is computationally more convenient. However, its validity for various models has not been carefully studied. We are aware of some cases in which first order approximation leads to spurious results. For example, when people compare welfare across alternative policies, non-linearity may matter, which makes first order approximation unreliable. Undoubtedly, when first order approximation is not sufficient to accurately approximate the model to the extent of our purposes, we need to resort to higher order approximation. We will generally deal with two main questions: the first is how higher order approximation should be accurately and effectively conducted and then what difference it makes when we apply higher order approximation. Clearly, this paper is far from sufficient to comprehensively answer all those questions. Instead we follow the approach proposed by Schmitt-Grohe & Uribe (2002) and apply the second order approximation to policy functions to solve the Hansen's (1985) real business cycle model, and compare the numerical results with those by the first order approximation.

Usually, we solve DSGE models by finding the decision rules from a nonlinear system of first order conditions and other conditions that in all characterize the equilibrium. One common way to pursue is to approximate the nonlinear system of equations around the non-stochastic steady state by Taylor's expansion at a certain order and then to find the recursive law of motion from the approximated system. (In the following, we will call this approach the usual approach.) There is plenty of literature on this topic. In the case of first order approximation, Campbell (1994) and Uhlig (1995) employ the method of undetermined coefficients to solve the neoclassical growth model. Blanchard & Kahn (1980) and Sims (2000a) propose a general procedure to derive the stable solution for a system of first order difference equations. For the case of second order approximation, Kim et al. (2003) would be the reference showing the solution method by second order approximation.

As a different approach, Schmitt-Grohe & Uribe (2002) propose to solve DSGE models by approximating the policy function at steady states. The unknown coefficients in approximated policy functions are the gradients of the policy function evaluated at steady states. They can be solved for from the fact that the differentiated non-linear system at any order with the policy function plugged in is equal to zero.

In this paper, we compare these two approaches in some depth and prove that in the case of first order approximation, the two approaches lead to the same equation-solving problem. Then we choose Hansen's real business cycle model as a standard model and follow the approach proposed by Schmitt-Grohe & Uribe (2002) to solve for the recursive law of motion at second order. Then we do the impulse response and simulation experiment with the second order recursive law of motion and find that the impulse response at second order converges to a new level due to the property of the second order recursive law of motion and the difference of impulse response between first order and second order is very small in the standard calibration but becomes larger for other values of relative risk aversion. The calculated second moments tend to be very close between first order and second order for all parameters tested.

The paper is organized as follows: In part 2, we have a brief review on some of the key articles on first order and second order approximation. We describe the settings of Hansen's real business cycle (RBC) model in part 3. In part 4, we first describe in detail the approach of approximation to the policy function by Schmitt-Grohe & Uribe (2002). Then we make a brief comparison between the approach of approximation to the policy function and the usual approach and prove that at first order the two approaches lead to the same system of equations to be solved. In the third section of part 4, we specifically solve the Hansen's RBC model. In part 5, we present the results. We test the model with different values of relative risk aversion in part 6. In part 7, we make some comments on the results and the two main general questions concerning second order approximation. Part 8 concludes.

2. Literature

It is worth spending some paragraphs on first order approximation, as it is the necessary step for higher order approximation. As a usual representation, we solve DSGE models for the recursive law of motion in percentage deviations of level variables. To that end, loglinearization is a common way to linearize the model and express the model in percentage deviations. Uhlig (1995) mentions the general procedure of loglinearization as expressing the model directly in log deviations, and then linearizing the model in log deviations by first order Taylor's expansion. As is known, log deviations are approximately percentage deviations. Besides, he proposes a simple procedure for loglinearization. That is to replace any variable X_t with $X_t = \overline{X}e^{\hat{x}_t}$, where $\overline{X}$ is the steady state and $\hat{x}_t = \log X_t - \log \overline{X}$ is the log deviation, approximately the percentage deviation from the steady state. Then $e^{\hat{x}_t}$ can be replaced by $1 + \hat{x}_t$.

Many authors loglinearize the model by first expressing the model in logs of the level variables and then linearizing the system of equations by Taylor's expansion. The linearized model is in logs of the level variables. So is the derived the recursive law of motion. And you can very easily transform the recursive law of motion into the one in log deviations. In fact, when we linearize the model in logs of the level variables by Taylor expansion, we can directly achieve the model in log deviations. And then solve the system in log deviations instead. It is worth noting that taking logs, though convenient, is not the only way to transform the linearized models in level variables into the ones in percentage deviations. For any equation, we can linearize its both sides by applying first order Taylor's expansion around steady states, and then for each variable X_t we can easily construct its percentage deviation as $\hat{x}_t = \frac{X_t - \overline{X}}{\overline{X}}$.

After the linearization of the nonlinear model, we reach a dynamic system of linear equations, from which we have to solve for the recursive law of motion. A number of authors have developed the solution methods for solving this system of linear difference equations. Campbell (1994) and Uhlig (1995) employ the method of undetermined coefficients. Uhlig (1995) states the system of difference equations in matrix expressions. So is the assumed recursive law of motion. Plugging in the assumed recursive law of motion results in solving a matrix quadratic equation, which can be reduced to a generalized eigenvalue problem. With the same mathematical mechanism, Blanchard & Kahn (1980) and Sims (2000a) propose a general procedure to derive the stable solution for a system of first order difference equations. Generally, they first formulate the economic model in the form of systems of first order difference equations. Then through QZ decomposition, generalized eigenvales are calculated. The difference equations corresponding to the eigenvalues less than one are to be solved backwards. Those corresponding to the eigenvalues more than one are to be solved forwards. One thing worth noting is that these general solution methods for systems of first order difference equations are crucial in that we also resort to these methods for calculating coefficients on linear terms when we solve in the following the DSGE models by second order approximation to the policy function.

Apart from Schmitt-Grohe & Uribe (2002), there are some other papers on second order approximation. For example, Kim et al (2003) also propose a solution algorithm to solve DSGE models by accurate second order approximation. They start to derive their solution methods from first applying second order Taylor's expansion to the whole system of equations. They also shed some light on forecasting and simulation, local accuracy of approximation and welfare comparison.

As an example where linear approximation may be unreliable, Kim & Kim (forthcoming) compare the welfare based on an evaluation of the utility function using a linear approximation to the policy function in a simple two-agent economy. They find

that the linear approximation to the policy function may lead to the spurious result that welfare is higher under autarky than full risk sharing.

Schmitt-Grohe & Uribe (2002) is the main reference which lays out a solution method for second order approximation to the policy function. We will present their approach in detail in part 4.

3. The Model

The RBC model to which we will apply second approximation is Hansen's (1985) RBC model. In this section, we briefly explain the model.

First, the social planner maximizes the representative agent's utility as

$$E\left[\sum_{t=0}^{\infty}\beta^t\left(\frac{C_t^{1-\eta}-1}{1-\eta}-AN_t\right)\right] \tag{1}$$

subject to

$$C_t + K_{t+1} = Z_t K_t^{\rho} N_t^{1-\rho} + (1-\delta)K_t \tag{2}$$

$$\log Z_t = (1-\psi)\log\overline{Z} + \psi \log Z_{t-1} + \varepsilon_t \tag{3}$$

$$\varepsilon_t \sim i.i.d.N(0;\sigma^2),$$

where E is the expectation operator; β, C_t, η, N_t and A are respectively the discount factor, consumption at time t, the coefficient of relative risk aversion, labor at time t, and the labor parameter; and K_t, Z_t, ρ, δ and ψ are respectively the capital stock at time t, technology at time t, the capital share, the depreciation rate and the autocorrelation

of technology; ε_t is the exogenous shock which is identically, independently normally distributed with standard deviation σ. Equation (2) is the budget constraint which also includes the production function, and equation (3) is the evolution of productivity.

The relevant equilibrium conditions are:

$$C_t + K_{t+1} = Z_t K_t^{\rho} N_t^{1-\rho} + (1-\delta)K_t \tag{2}$$

$$C_t^{-\eta} = E_t\left[\beta C_{t+1}^{-\eta}(\rho Z_{t+1} K_{t+1}^{\rho-1} N_{t+1}^{1-\rho} + (1-\delta))\right] \tag{4}$$

$$A = C_t^{-\eta}(1-\rho) Z_t K_t^{\rho} N_t^{-\rho} \tag{5}$$

$$\log Z_{t+1} = (1-\psi)\log \overline{Z} + \psi \log Z_t + \varepsilon_{t+1} \tag{3}$$

$$Y_t = Z_t K_t^{\rho} N_t^{1-\rho} \tag{6}$$

$$R_t = \rho \frac{Y_t}{K_t} + (1-\delta) \tag{7}$$

In the above equations, (4) is the first order condition for capital and (5) is the first order condition for labor. (6) and (7) are the production function and the expression for the interest rate respectively.

4. Model Analysis

4.1 The approach of approximation to the policy function

To solve the model by second order approximation to the policy function, we follow the approach proposed by Schmitt-Grohe & Uribe (2002). We briefly repeat the main procedure and arguments of their approach. They formulate the set of equilibrium conditions of usual macroeconomic models as

$$E_t f(y_{t+1}, y_t, x_{t+1}, x_t) = 0, \tag{8}$$

where E_t is the mathematical expectation operator conditional on information up to time t. y_t , of size $n_y \times 1$, is the vector of co-state variables and x_t, of size $n_x \times 1$, is the vector of state variables which include predetermined state variables and exogenous state variables. It is defined that $n = n_y + n_x$. The function f maps $R^{n_y} \times R^{n_y} \times R^{n_x} \times R^{n_x}$ into R^n. x_t can be partitioned as

$$x_t = \left[x_t^1; x_t^2\right]',$$

where the vector x_t^1 consists of endogenous predetermined state variables and the vector x_t^2 consists of exogenous state variables. x_t^2 is assumed to follow the exogenous stochastic process given by

$$x_{t+1}^2 = \Lambda x_t^2 + \tilde{\theta}\sigma\varepsilon_{t+1}; \tag{9}$$
$$\varepsilon_t \sim i.i.d.N(0, I).$$

The vector x_t^2 and the innovation ε_t are both of size $n_\varepsilon \times 1$. The vector ε_t is independently, identically and normally distributed with mean zero and covariance matrix I. $\tilde{\theta}$ is of size $n_\varepsilon \times n_\varepsilon$ and consists of known parameters. $\sigma \geq 0$ and is in fact a scaling parameter which scales the variance of the innovation. All eigenvalues of the matrix Λ are assumed to be within the unit circle.

Then the solution to the model, the so-called policy function, is specified as the following general form:

$$y_t = g(x_t, \sigma) \tag{10}$$

$$x_{t+1} = h(x_t, \sigma) + \theta\sigma\varepsilon_{t+1}, \tag{11}$$

where g maps $R^{n_x} \times R^{+}$ into R^{n_y} and h maps $R^{n_x} \times R^{+}$ into R^{n_x}. θ is a $n_x \times n_\varepsilon$ matrix and is given by

$$\theta = \begin{bmatrix} 0 \\ \widetilde{\theta} \end{bmatrix}.$$

Since it is formidably difficult to derive the true form of the policy function, the best to do is to approximate the policy function at a certain order around the non-stochastic steady states $x_t = \bar{x}$ and $\sigma = 0$. That is the central idea of this approach. And the non-stochastic steady states are defined as vectors $(\bar{x}, \bar{y})$ such that

$$f(\bar{y}, \bar{y}, \bar{x}, \bar{x}) = 0. \tag{12}$$

By applying Taylor's expansion, the policy functions are approximated at first order as follows:

$$g(x, \sigma) = g(\bar{x}, 0) + g_x(\bar{x}, 0)(x - \bar{x}) + g_\sigma(\bar{x}, 0)\sigma \tag{13}$$

$$h(x, \sigma) = h(\bar{x}, 0) + h_x(\bar{x}, 0)(x - \bar{x}) + h_\sigma(\bar{x}, 0)\sigma. \tag{14}$$

From the above equations, we drop time subscripts and use a prime to indicate variables dated in time $t+1$.

At second order, the approximated policy function is given by

$$\begin{aligned} [g(x,\sigma)]^i = {} & [g(\bar{x},0)]^i + [g_x(\bar{x},0)]^i_a [(x-\bar{x})]_a + [g_\sigma(\bar{x},0)]^i [\sigma] \\ & + \frac{1}{2}[g_{xx}(\bar{x},0)]^i_{ab} [(x-\bar{x})]_a [(x-\bar{x})]_b \end{aligned}$$

$$+\frac{1}{2}\left[g_{x\sigma}(\bar{x},0)\right]^{i}_{a}\left[(x-\bar{x})\right]_{a}\left[\sigma\right]$$

$$+\frac{1}{2}\left[g_{\sigma x}(\bar{x},0)\right]^{i}_{a}\left[(x-\bar{x})\right]_{a}\left[\sigma\right]+\frac{1}{2}\left[g_{\sigma\sigma}(\bar{x},0)\right]^{i}\left[\sigma\right]\left[\sigma\right] \tag{15}$$

$$\left[h(x,\sigma)\right]^{j}=\left[h(\bar{x},0)\right]^{j}+\left[h_{x}(\bar{x},0)\right]^{j}_{a}\left[(x-\bar{x})\right]_{a}+\left[h_{\sigma}(\bar{x},0)\right]^{j}\left[\sigma\right]$$

$$+\frac{1}{2}\left[h_{xx}(\bar{x},0)\right]^{j}_{ab}\left[(x-\bar{x})\right]_{a}\left[(x-\bar{x})\right]_{b}$$

$$+\frac{1}{2}\left[h_{x\sigma}(\bar{x},0)\right]^{j}_{a}\left[(x-\bar{x})\right]_{a}\left[\sigma\right]$$

$$+\frac{1}{2}\left[h_{\sigma x}(\bar{x},0)\right]^{j}_{a}\left[(x-\bar{x})\right]_{a}\left[\sigma\right]+\frac{1}{2}\left[h_{\sigma\sigma}(\bar{x},0)\right]^{j}\left[\sigma\right]\left[\sigma\right], \tag{16}$$

where $i=1,...,n_y$ and $a,b,j=1,...,n_x$. And here we use the same tenor notation as in Schmitt-Grohe & Uribe (2002). For example, $\left[g_{xx}(\bar{x},0)\right]$ is an $n_y \times n_x \times n_x$ matrix and $\left[g_{xx}(\bar{x},0)\right]^{i}_{ab}$ is the (i,a,b) element of the second derivative of g with respect to x.

So the next step is to get the values of the unknowns in the approximated policy functions, whether at first order or second order. We find that the unknowns are the gradients of the policy functions evaluated at steady states. To solve for these unknowns, we have to resort to the system of equilibrium conditions. After plugging the policy functions into the system, we define the new system as

$$F(x,\sigma)\equiv E_t f(g(h(x,\sigma)+\theta\sigma\varepsilon',\sigma),g(x,\sigma),h(x,\sigma)+\theta\sigma\varepsilon',x\ )=0. \tag{17}$$

Since $F(x,\sigma)$ is equal to zero for any values of x and σ, the derivatives of any order of $F(x,\sigma)$ with respect to x and σ are equal to zero, which provides information for solving for the unknown coefficients in the approximated policy functions.

For the first order case, we have $F_x(\bar{x},0)=0$ and $F_\sigma(\bar{x},0)=0$. Specifically we have

$$\begin{aligned}[F_x(\bar{x},0)]^i_j &= [f_{y'}]^i_\alpha [g_x]^\alpha_\beta [h_x]^\beta_j + [f_y]^i_\alpha [g_x]^\alpha_j + [f_{x'}]^i_\beta [h_x]^\beta_j + [f_x]^i_j \\ &= 0; \end{aligned} \tag{18}$$

$$\begin{aligned}[F_\sigma(\bar{x},0)]^i &= E_t\{[f_{y'}]^i_\alpha [g_x]^\alpha_\beta [h_\sigma]^\beta + [f_{y'}]^i_\alpha [g_x]^\alpha_\beta [\theta]^\beta_\phi [\varepsilon']^\phi + [f_{y'}]^i_\alpha [g_\sigma]^\alpha + [f_y]^i_\alpha [g_\sigma]^\alpha \\ &\quad + [f_{x'}]^i_\beta [h_\sigma]^\beta + [f_{x'}]^i_\beta [\theta]^\beta_\phi [\varepsilon']^\phi\} \\ &= [f_{y'}]^i_\alpha [g_x]^\alpha_\beta [h_\sigma]^\beta + [f_{y'}]^i_\alpha [g_\sigma]^\alpha + [f_y]^i_\alpha [g_\sigma]^\alpha + [f_{x'}]^i_\beta [h_x]^\beta \\ &= 0; \end{aligned} \tag{19}$$

where $i=1,...,n$; $j,\beta=1,...,n_x$; $\alpha=1,...,n_y$ and $\phi=1,...,n_\varepsilon$. (18) is a system of $n\times n_x$ quadratic equations in the $n\times n_x$ unknowns constructed by the elements of g_x and h_x.(All the derivatives without apparent notation are also values evaluated at steady states.) A number of authors have developed methods to solve for g_x and h_x which lead to non-explosive paths for the state and control variables (e.g. Blanchard and Kahn (1980), Sims (2000a)). And g_σ and h_σ can be solved for from equation (19). Since equation (19) is linear and homogeneous in g_σ and h_σ, if a unique solution exists, it must be true that

$$h_\sigma = 0,$$

and

$$g_\sigma = 0.$$

That is one of the main theoretical results of Schmitt-Grohe & Uribe (2002), which states that generally, the size of variance of the shocks has no effect on the constant term of the approximation to the policy function up to first order. It implies that in first

order approximation the expected values of x_t and y_t are equal to their non-stochastic steady-state values $\bar{x}$ and $\bar{y}$.

Now we turn to the second order case. From the expressions of second order approximation to policy functions (15) and (16), we see that the coefficients of first order part g_x and h_x are already known from the first order case and the rest unknown coefficients are $[g_{xx}(\bar{x},0)]^i_{ab}$, $[g_{x\sigma}(\bar{x},0)]^i_a$, $[g_{\sigma x}(\bar{x},0)]^i_a$, $[g_{\sigma\sigma}(\bar{x},0)]^i$, $[h_{xx}(\bar{x},0)]^j_{ab}$, $[h_{x\sigma}(\bar{x},0)]^j_a$, $[h_{\sigma x}(\bar{x},0)]^j_a$, $[h_{\sigma\sigma}(\bar{x},0)]^j$. Those coefficients can be solved for by taking the derivative of $F(x,\sigma)$ with respect to x and σ twice and evaluating them at $(x,\sigma)=(\bar{x},0)$. First, from $F(x,\sigma)$ we can calculate $g_{xx}(\bar{x},0)$ and $h_{xx}(\bar{x},0)$. We have $[F_{xx}(\bar{x},0)]^i_{jk}$ as:

$$
\begin{aligned}
[F_{xx}(\bar{x},0)]^i_{jk} = &\left(\left[f_{y'y'}\right]^i_{\alpha\gamma}[g_x]^\gamma_\delta[h_x]^\delta_k + \left[f_{y'y}\right]^i_{\alpha\gamma}[g_x]^\gamma_k + \left[f_{y'x'}\right]^i_{\alpha\delta}[h_x]^\delta_k + \left[f_{y'x}\right]^i_{\alpha k}\right)[g_x]^\alpha_\beta[h_x]^\beta_j \\
&+ \left[f_{y'}\right]^i_\alpha[g_{xx}]^\alpha_{\beta\delta}[h_x]^\delta_k[h_x]^\beta_j + \left[f_{y'}\right]^i_\alpha[g_x]^\alpha_\beta[h_{xx}]^\beta_{jk} \\
&+ \left(\left[f_{yy'}\right]^i_{\alpha\gamma}[g_x]^\gamma_\delta[h_x]^\delta_k + \left[f_{yy}\right]^i_{\alpha\gamma}[g_x]^\gamma_k + \left[f_{yx'}\right]^i_{\alpha\delta}[h_x]^\delta_k + \left[f_{yx}\right]^i_{\alpha k}\right)[g_x]^\alpha_j \\
&+ \left[f_y\right]^i_\alpha[g_{xx}]^\alpha_{jk} \\
&+ \left(\left[f_{x'y'}\right]^i_{\beta\gamma}[g_x]^\gamma_\delta[h_x]^\delta_k + \left[f_{x'y}\right]^i_{\beta\gamma}[g_x]^\gamma_k + \left[f_{x'x'}\right]^i_{\beta\delta}[h_x]^\delta_k + \left[f_{x'x}\right]^i_{\beta k}\right)[h_x]^\beta_j \\
&+ \left[f_{x'}\right]^i_\beta[h_{xx}]^\beta_{jk} \\
&+ \left[f_{xy'}\right]^i_{j\gamma}[g_x]^\gamma_\delta[h_x]^\delta_k + \left[f_{xy}\right]^i_{j\gamma}[g_x]^\gamma_k + \left[f_{xx'}\right]^i_{j\delta}[h_x]^\delta_k + \left[f_{xx}\right]^i_{jk} \\
&= 0, \qquad (20)
\end{aligned}
$$

where $i=1,...,n$; $j,k,\beta,\delta=1,...,n_x$; and $\alpha,\gamma=1,...,n_y$.

The above expression is a system of $n\times n_x\times n_x$ linear equations in the $n\times n_x\times n_x$ unknowns which are the elements of g_{xx} and h_{xx}.

And from $F_{\sigma\sigma}(\bar{x},0)=0$ we can solve for $g_{\sigma\sigma}$ and $h_{\sigma\sigma}$. We have $F_{\sigma\sigma}(\bar{x},0)$ as :

$$
\begin{aligned}
\left[F_{\sigma\sigma}(\bar{x},0)\right]^i = & \left[f_{y'}\right]^i_\alpha \left[g_x\right]^\alpha_\beta \left[h_{\sigma\sigma}\right]^\beta + \left[f_{y'y'}\right]^i_{\alpha\gamma} \left[g_x\right]^\gamma_\delta \left[\theta\right]^\delta_\xi \left[g_x\right]^\alpha_\beta \left[\theta\right]^\beta_\phi \left[I\right]^\phi_\xi \\
& + \left[f_{y'x'}\right]^i_{\alpha\delta} \left[\theta\right]^\delta_\xi \left[g_x\right]^\alpha_\beta \left[\theta\right]^\beta_\phi \left[I\right]^\phi_\xi \\
& + \left[f_{y'}\right]^i_\alpha \left[g_x\right]^\alpha_{\beta\delta} \left[\theta\right]^\delta_\xi \left[\theta\right]^\beta_\phi \left[I\right]^\phi_\xi + \left[f_{y'}\right]^i_\alpha \left[g_{\sigma\sigma}\right]^\alpha + \left[f_y\right]^i_\alpha \left[g_{\sigma\sigma}\right]^\alpha \\
& + \left[f_{x'}\right]^i_\beta \left[h_{\sigma\sigma}\right]^\beta \\
& + \left[f_{x'y'}\right]^i_{\beta\gamma} \left[g_x\right]^\gamma_\delta \left[\theta\right]^\delta_\xi \left[\theta\right]^\beta_\phi \left[I\right]^\phi_\xi + \left[f_{x'x'}\right]^i_{\beta\delta} \left[\theta\right]^\delta_\xi \left[\theta\right]^\beta_\phi \left[I\right]^\phi_\xi \\
& = 0 , \qquad (21)
\end{aligned}
$$

where $i = 1,...,n$; $\beta, \delta = 1,...,n_x$; $\alpha, \gamma = 1,...,n_y$; $\phi, \xi = 1,...,n_\varepsilon$ and I is the covariance matrix of the innovation. It is a system of n linear equations in the n unknowns given by the elements of $g_{\sigma\sigma}$ and $h_{\sigma\sigma}$.

The last step is to solve for the cross derivatives $g_{x\sigma}(\bar{x},0)$ and $h_{x\sigma}(\bar{x},0)$. From $h_\sigma = 0$ and $g_\sigma = 0$, we can write $\left[F_{\sigma x}(\bar{x},0)\right]$ as:

$$
\begin{aligned}
\left[F_{\sigma x}(\bar{x},0)\right]^i_j = & \left[f_{y'}\right]^i_\alpha \left[g_x\right]^\alpha_\beta \left[h_{\sigma x}\right]^\beta_j + \left[f_{y'}\right]^i_\alpha \left[g_{\sigma x}\right]^\alpha_\gamma \left[h_x\right]^\gamma_j + \left[f_y\right]^i_\alpha \left[g_{\sigma x}\right]^\alpha_j + \left[f_{x'}\right]^i_\beta \left[h_{\sigma x}\right]^\beta_j \\
& = 0 , \qquad (22)
\end{aligned}
$$

where $i = 1,...,n$; $\beta, \gamma, j = 1,...,n_x$; and $\alpha = 1,...,n_y$.

It is a system of $n \times n_x$ equations in the $n \times n_x$ unknowns which are the elements of $g_{\sigma x}$ and $h_{\sigma x}$. Since the system is homogenous in the unknowns, if a unique solution exists, it is given by

$$g_{\sigma x} = 0 ,$$

and

$h_{\sigma x} = 0$.

Again, these equations are another main theoretical result of Schmitt-Grohe & Uribe (2002). They imply that generally, up to second order, the size of the variance of the shocks does not affect the coefficients of the policy function on the terms that are linear in the state vector.

4.2 Comparison of Two Approaches

Before we proceed, it is worth making a comparison between the two approaches, as the above approach by Schmitt-Grohe & Uribe (2002) is quite different from the one people usually employ. We can see that at the first order approximation, the two approaches finally lead to the same system of equations to be solved.

For first order approximation, by the usual approach, people first approximate the nonlinear system of equations around steady states by Taylor's expansion at first order and then find the recursive law of motion from the approximated system. By the Uhlig (1995)'s method of undetermined coefficients, the linear recursive law of motion is assumed, it is plugged into the linearized system of equations and the unknown coefficients are solved for. Now by comparison, we can see these two approaches are virtually doing the same thing at first order case. By the approach by Schmitt-Grohe & Uribe (2002), we first approximated the policy function at first order. And approximated policy functions are the same as the assumed recursive law of motion in the usual approach, also with the same unknown coefficients when transformed in a proper way. Then by the approach of Schmitt-Grohe & Uribe (2002), we plug the policy function into the system and calculate the unknown coefficients by differentiating the system with respect to x and σ . This step virtually functions the same as, by the usual approach, plugging the assumed recursive law of motion into the linearized system and solve for the unknown coefficients from setting the sums of coefficients of each variable

to zero. Finally, these two approaches result in the same equations to be solved. (Please see a brief proof in Appendix a).

4.3 Solving the RBC model

Now we specifically apply Schmitt-Grohe & Uribe (2002)'s method to solve Hansen (1985)'s model. First we define the vectors y and x for our model as:

$$y = [c] = [\log C]$$

and

$$x = \begin{bmatrix} k \\ z \end{bmatrix} = \begin{bmatrix} \log K \\ \log Z \end{bmatrix}.$$

We include only three variables in the dynamic system and express them in logs. c , the consumption, is the control variable and belongs to the vector y . k , the capital stock, and z , technology, are the predetermined state variable and the exogenous state variable respectively, and belong to the vector x . And the system includes the following three equations:

$$\exp(c) + \exp(k') = \exp(z)\exp(k)^{\rho}\left(\left(\frac{(1-\rho)\exp(c)^{-\eta}\exp(z)\exp(k)^{\rho}}{A}\right)^{\frac{1}{\rho}}\right)^{1-\rho} + (1-\delta)\exp(k), \tag{24}$$

$$\exp(c)^{-\eta} = E_t\left[\beta\exp(c')^{-\eta}\left(\rho\exp(z')\exp(k')^{\rho-1}\left(\left(\frac{(1-\rho)\exp(c')^{-\eta}\exp(z')\exp(k')^{\rho}}{A}\right)^{\frac{1}{\rho}}\right)^{1-\rho} + (1-\delta)\right)\right], \tag{25}$$

$$z' = \psi z + \sigma\varepsilon' . \tag{26}$$

f consists of those three equations. Equations (24) and (25) are derived by plugging equation (5) into equations (2) and (4) and expressing level variables with their logs; equation (26) is equation (3) expressed in logs of level variables with $\overline{Z} = 1$.

We resort to a package of Matlab programs mainly by Schmitt-Grohe & Uribe (2002) to solve for the coefficients of the approximated policy function. In the program package, we specifiy the settings of our model mainly in the programs RBC_model_ss.m and RBC_model.m. (Please see the Appendix b for detail).

The advantage of expressing the system in logs is that the approximated policy functions we get are the same as the recursive law of motion in percentage deviations. When we obtain the recursive law of motion for c, k and z, all the other variables will be respectively calculated from the relevant equilibrium conditions. Specifically we have:

$$N = \left(\frac{(1-\rho)C^{-\eta} Z K^{\rho}}{A} \right)^{\frac{1}{\rho}}, \tag{27}$$

$$Y = ZK^{\rho} N^{1-\rho},$$

$$R = \rho \frac{Y}{K} + (1-\delta).$$

For computational convenience, especially in simulation, we linearize the above equations and express them in percentage deviations.

$$\hat{n} = -\frac{\eta}{\rho}\hat{c} + \hat{k} + \frac{1}{\rho}\hat{z} \tag{28}$$

$$\hat{y} = (1-\rho)\hat{n} + \rho\hat{k} + \hat{z}, \tag{29}$$

$$\hat{r} = \frac{\rho \overline{Y}}{\overline{KR}} (\hat{y} - \hat{k}) . \tag{30}$$

5. Results

We follow the calibration in Hansen (1985), which sets $\beta = 0.99$, $\rho = 0.36$, $\delta = 0.0025$, $\psi = 0.95$, $\eta = 1$, $\sigma = 0.00712$, $\overline{Z} = 1$ and $\overline{N} = \frac{1}{3}$.

The coefficients of linear terms are

$$g_x = \begin{bmatrix} 0.5315 & 0.4696 \end{bmatrix}$$

$$h_x = \begin{bmatrix} 0.9420 & 0.1550 \\ -0.0000 & 0.9500 \end{bmatrix}.$$

The coefficients of the quadratic terms are:

$$g_{xx}(:,:,1) = \begin{bmatrix} 0.0593 & -0.1428 \end{bmatrix}$$
$$g_{xx}(:,:,2) = \begin{bmatrix} -0.1428 & 0.2487 \end{bmatrix}$$

$$h_{xx}(:,:,1) = \begin{bmatrix} 0.0531 & -0.1186 \\ 0 & 0 \end{bmatrix}$$
$$h_{xx}(:,:,2) = \begin{bmatrix} -0.1186 & 0.2661 \\ 0 & 0 \end{bmatrix},$$

where g_{xx} and h_{xx} are both three dimensional matrices.

The coefficients of the quadratic terms in σ are:

$$g_{\sigma\sigma} = -0.3148$$

$$h_{\sigma\sigma} = \begin{bmatrix} 0.0771 \\ 0 \end{bmatrix}.$$

So we have obtained the following recursive law of motion:

At first order:

$$\hat{c} = 0.5315\hat{k} + 0.4696\hat{z}$$

$$\hat{k}' = 0.9420\hat{k} + 0.1550\hat{z}\ .$$

At second order:

$$\hat{c} = 0.5315\hat{k} + 0.4696\hat{z} + \frac{1}{2}\left[0.0593\hat{k}^2 - 0.2856\hat{k}\hat{z} + 0.2487\hat{z}^2 - 0.3148\sigma^2\right]$$

$$\hat{k}' = 0.9420\hat{k} + 0.1550\hat{z} + \frac{1}{2}\left[0.0531\hat{k}^2 - 0.2372\hat{k}\hat{z} + 0.2661\hat{z}^2 + 0.0771\sigma^2\right].$$

We see that the coefficients of linear terms are the same for both cases. And they are exactly the same as those calculated by Uhlig's (1995) toolkit programs. Compared with first order approximation, second order approximation generates some quadratic terms and the additional constant term that is the quadratic term of scaling parameter, namely the standard deviation of the innovation. So we see that at first order approximation, the volatility of shocks does not matter for the decision rule, however, it does at second order approximation. That means at second order approximation, the economic agent will take into account the volatility of uncertainties when he makes decisions.

Now we calculate and compare impulse responses of all variables to a one standard

deviation shock to technology at first order and second order. The number of time periods is 100. For consumption and capital, the impulse responses are to be calculated from above recursive laws of motion. Based on recursive laws of motion for capital and consumption, we calculate labor, output, and interest rate from equations (28)-(30). Figures 1&2 show, at first order and second order respectively, the impulse responses of all variables to a one standard deviation shock to technology at time 2. Figure 1 shows the same impulse response as the one done by Uhlig's (1995) toolkit programs except that the capital stock is shifted one time period forward in our graph due to its different time notation. We see Figures 1&2 are generally the same without any sharp difference.

However, when we compare the individual series in details we can see the difference of their evolutions. Figures 3 to 7 individually show the impulse response of each variable-- capital, consumption, output, labor and interest respectively. All graphs on the right hand in each figure show the evolutions of each variable both at first order and second order; all graphs on the left show their difference (second order minus first order). We can see that the difference for each variable is very small so that in all graphs on the right hand the two impulse response curves almost become one. Figure 1 shows that the capital stock in both cases has a hump shape response, however their difference does not converge to zero but to another very small number. That means that the capital stock at second order does not converge to zero, because we know the impulse response at first order converges to zero. This is the case for all the other variables in Figures 4-7. In Figure 4, the consumption also has a hump shape response and their difference goes down a bit in the negative value, and then rises up and converges to about 0.00005. Figure 5 shows that the difference in the output goes up and converges to about 0.000016. And Figures 6&7, the difference in the labor and the interest rate converges to about 0.000006 and -0.0000006 respectively

So we find that in our standard calibration, the difference of impulse responses between first order and second order is very small and at second order the impulse response does not converge to zero but to another value. We can see that in the second order recursive

Figure 1

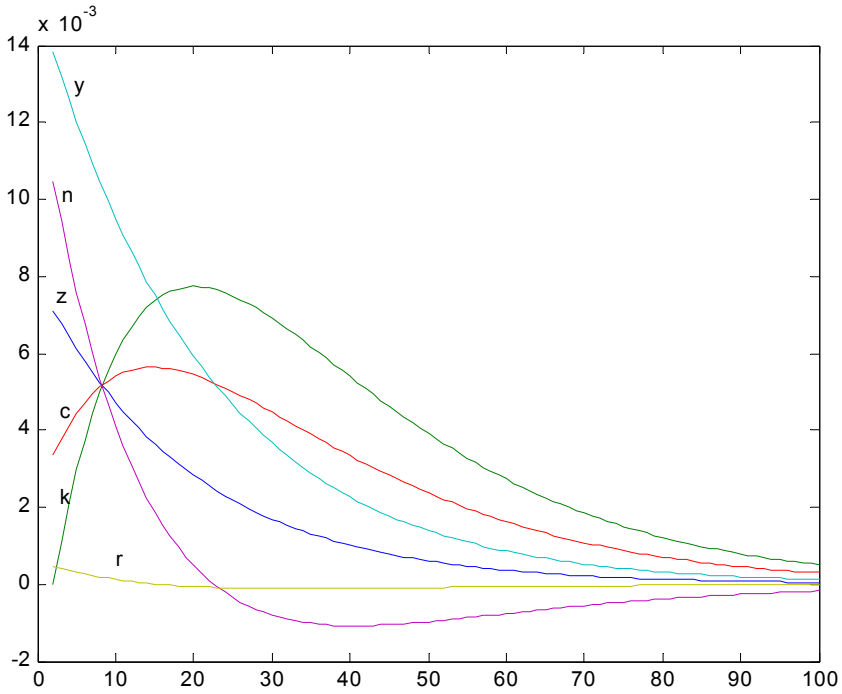

Figure 2

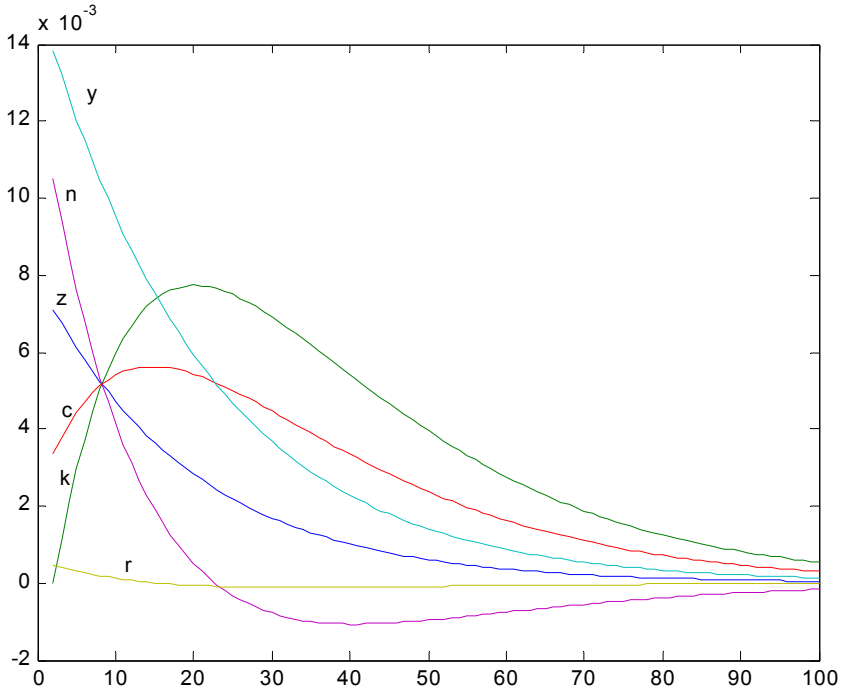

Figure3

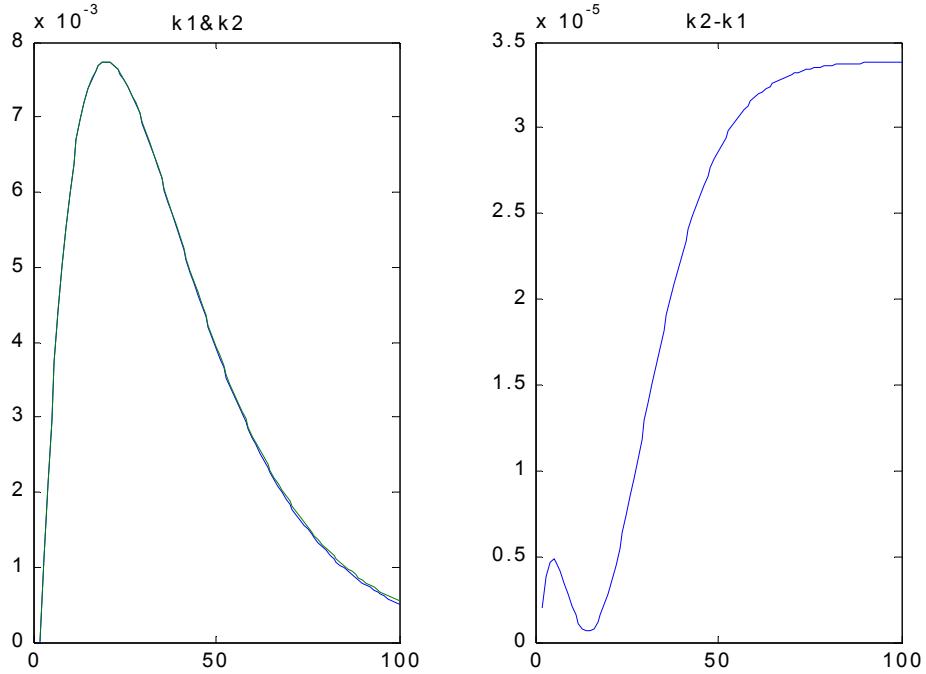

Figure4

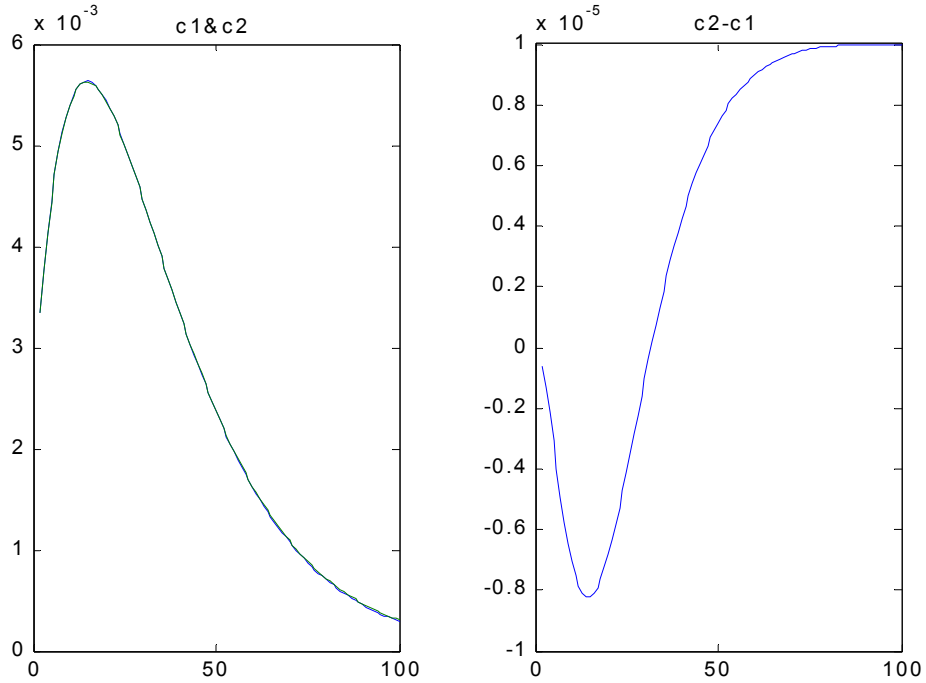

Figure5

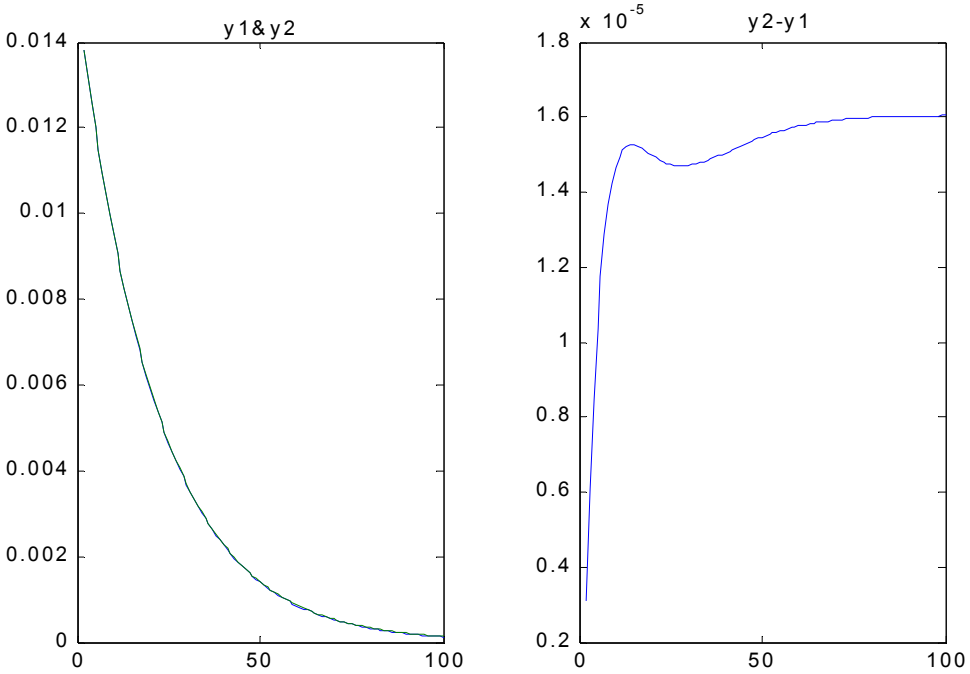

Figure6

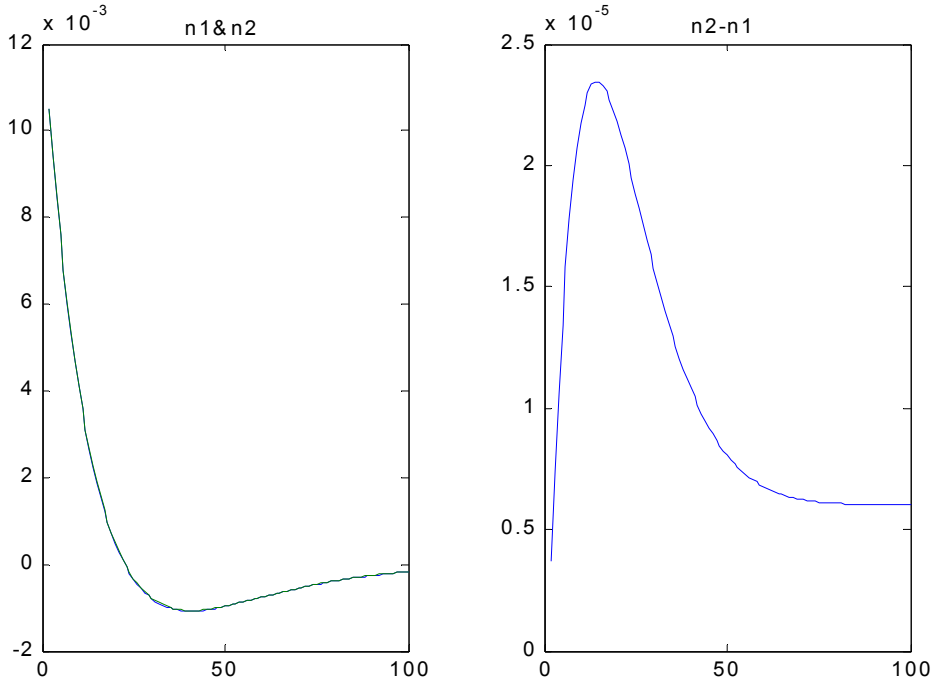

Figure 7

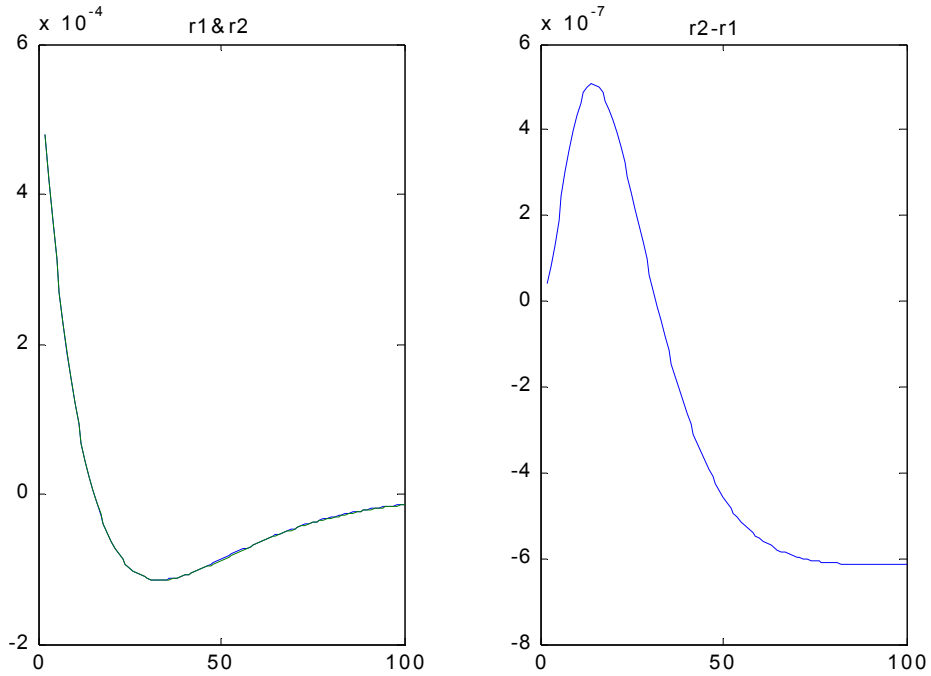

Table 1

	First order		Second order	
	Standard Deviation	Correlation with output	Standard Deviation	Correlation with output
Technology	0.0217	0.9993	0.0218	0.9993
Capital	0.0403	0.7021	0.0406	0.7043
Consumption	0.0294	0.8641	0.0296	0.8648
Labor	0.0234	0.7826	0.0235	0.7809
Output	0.0438	1	0.0440	1
Interest	0.0011	0.4714	0.0011	0.4677

law of motion, there is the quadratic term of scaling parameter, which is the constant term in the law of motion, and it enters the law of motion at each time and causes each variable to converge to another value rather than zero.

Now let us turn to the second moments calculated from the simulated series. The time length is 600 and each variable is simulated for 1000 times. The variables are simulated directly from the recursive law of motion and the series are not filtered by HP filter before the second moments are calculated. Table 1 shows the standard deviation of each variable and its correlation with the output at first order and second order. We find that the second moments are very close between the two cases and some of them are even the same at our rounding level.

6. Variation

In this section, we solve the model and calculate impulse responses and second moments with two different values of the coefficient of relative risk aversion. Rather than $\eta = 1$ in our standard calibration, we test the model with $\eta = 0.1$ and $\eta = 10$ respectively. And we have the following calculated recursive law of motion for the consumption and the capital stock.

$\eta = 0.1$

At first order:

$\hat{c} = 1.4552\hat{k} - 3.8323\hat{z}$

$\hat{k}' = 0.9420\hat{k} + 0.6131\hat{z}$.

At second order:

$$\hat{c} = 1.4552\hat{k} - 3.8323\hat{z} + \frac{1}{2}\left[-0.1943\hat{k}^2 + 2.6670\hat{k}\hat{z} - 11.7820\hat{z}^2 - 248.3524\sigma^2\right]$$

$$\hat{k}' = 0.9420\hat{k} + 0.6131\hat{z} + \frac{1}{2}\left[0.0056\hat{k}^2 - 0.0896\hat{k}\hat{z} + 0.7813\hat{z}^2 + 22.2291\sigma^2\right].$$

$\eta = 10$

At first order:

$$\hat{c} = 0.0723\hat{k} + 0.0894\hat{z}$$
$$\hat{k}' = 0.9420\hat{k} + 0.1092\hat{z}\ .$$

At second order:

$$\hat{c} = 0.0723\hat{k} + 0.0894\hat{z} + \frac{1}{2}\left[0.0123\hat{k}^2 - 0.0544\hat{k}\hat{z} + 0.0027\hat{z}^2 - 0.9719\sigma^2\right]$$

$$\hat{k}' = 0.9420\hat{k} + 0.1092\hat{z} + \frac{1}{2}\left[0.0732\hat{k}^2 - 0.1748\hat{k}\hat{z} + 0.1201\hat{z}^2 + 1.7500\sigma^2\right].$$

By the above calculated recursive law of motion, we calculated impulse responses and second moments from the simulated series. Figures 8-12 and Table 2 show the impulse responses and second moments for $\eta = 0.1$, and Figure 13-17 and Table 3 for $\eta = 10$. So when $\eta = 0.1$, we see that the difference of impulse responses of each variable between first order and second order becomes relatively large. In Figure 8, the capital stock at second order rise quite higher than at first order, and the difference converges to about 0.01. In Figure 9, the consumption at second order is first below the one at first order and then catches up and becomes higher, and the difference converges to about 0.008. In Figures 10&11, the output and the labor at second order rise up quite higher than at first order and the difference converges to about 0.008 and 0.0075 respectively. However, in Figure 12, the interest rate at second order first stays higher and then goes

Figure 8

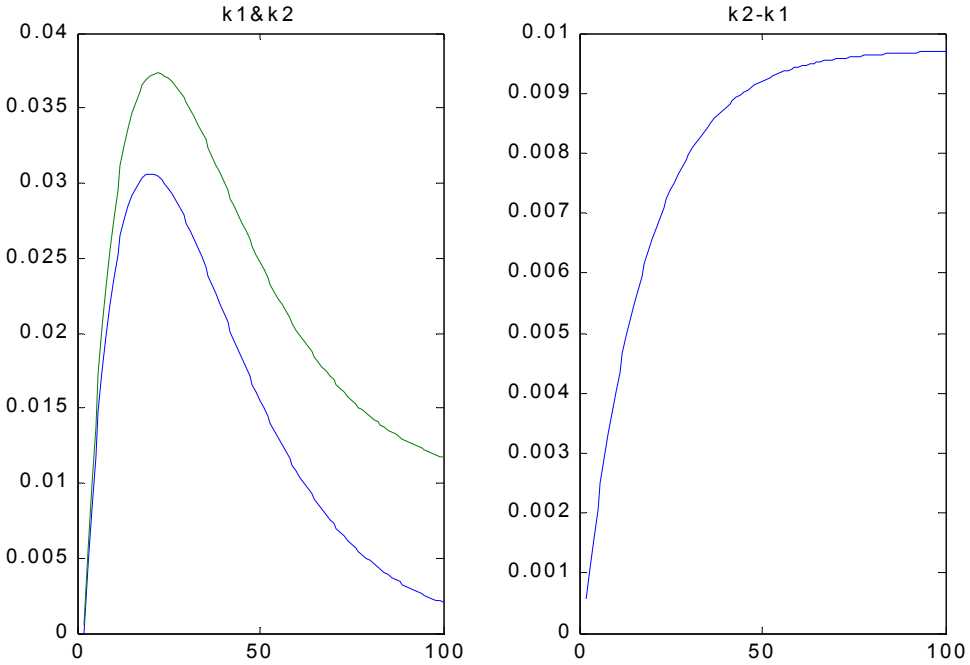

Figure 9

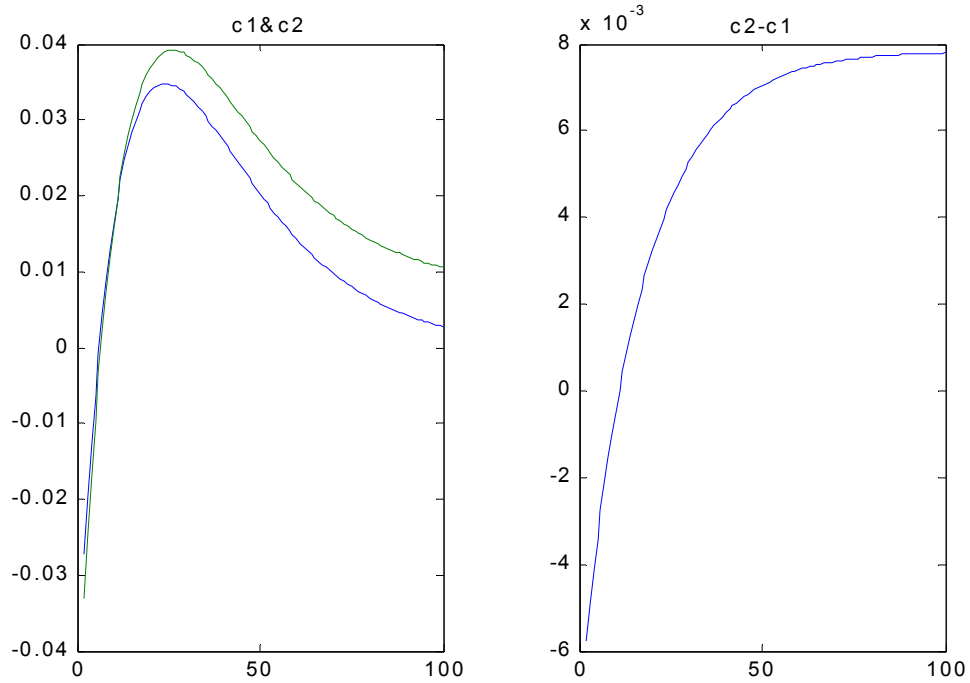

Figure 10

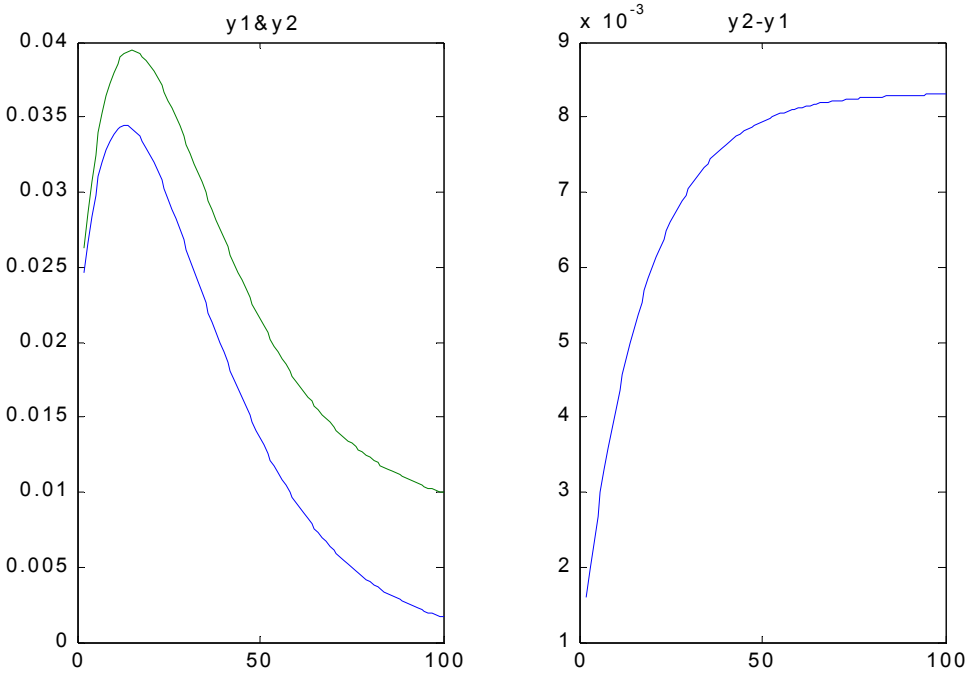

Figure 11

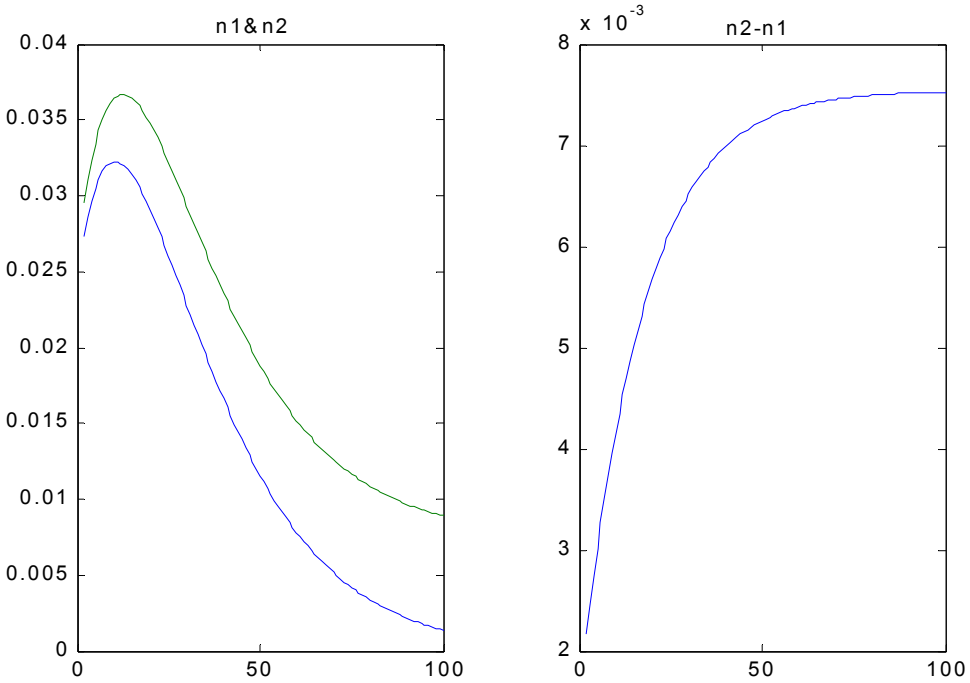

Figure 12

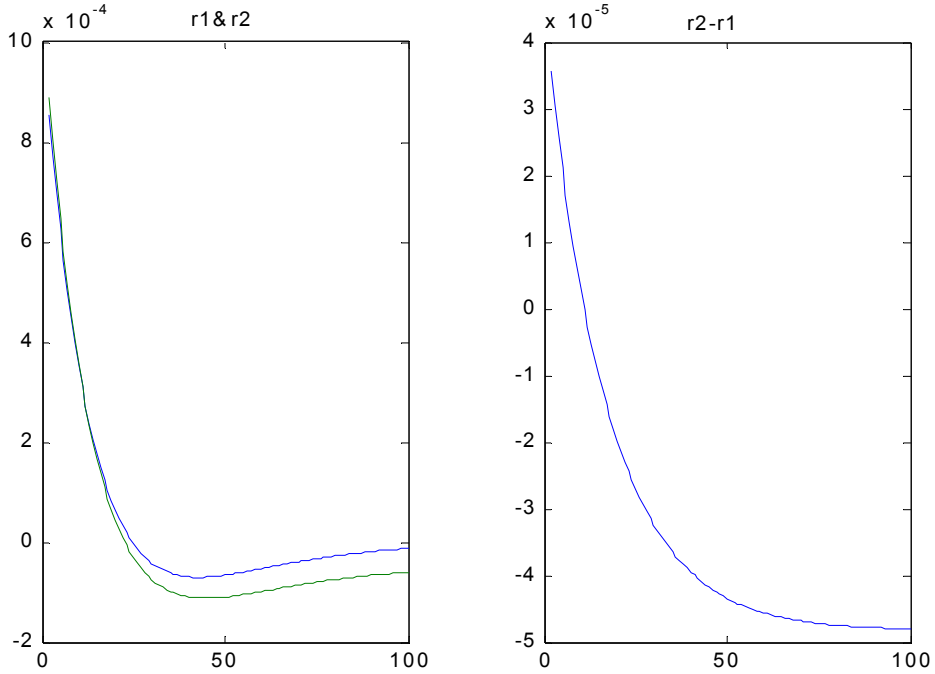

Figure13

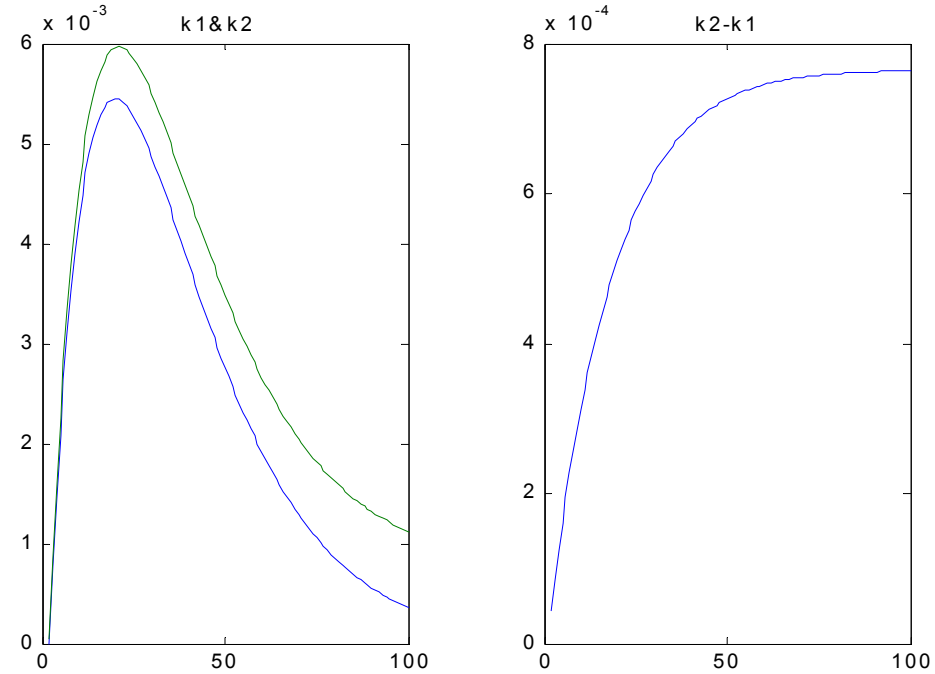

Figure14

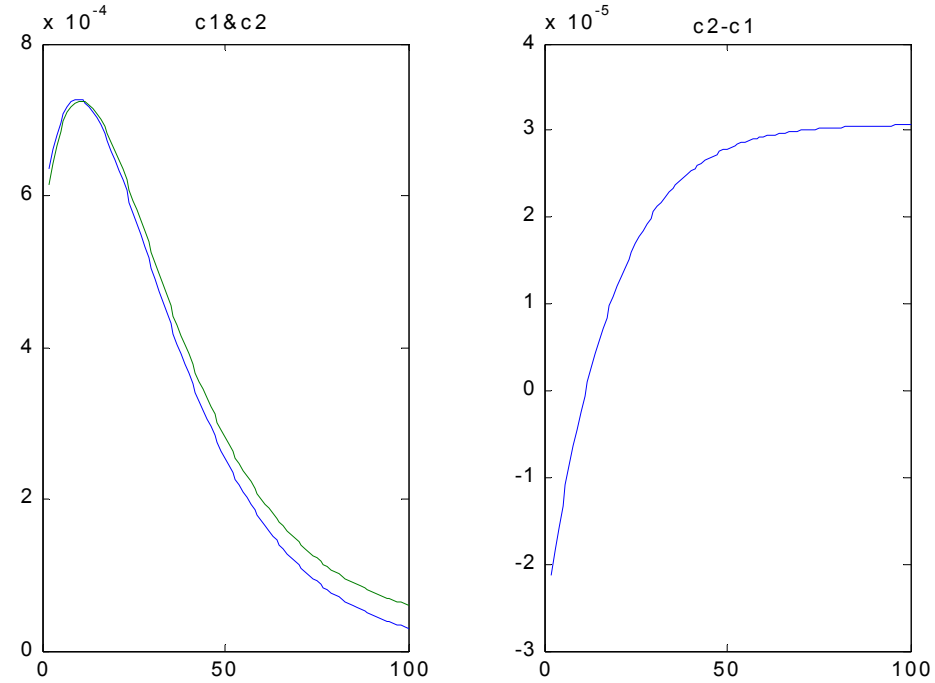

Figure15

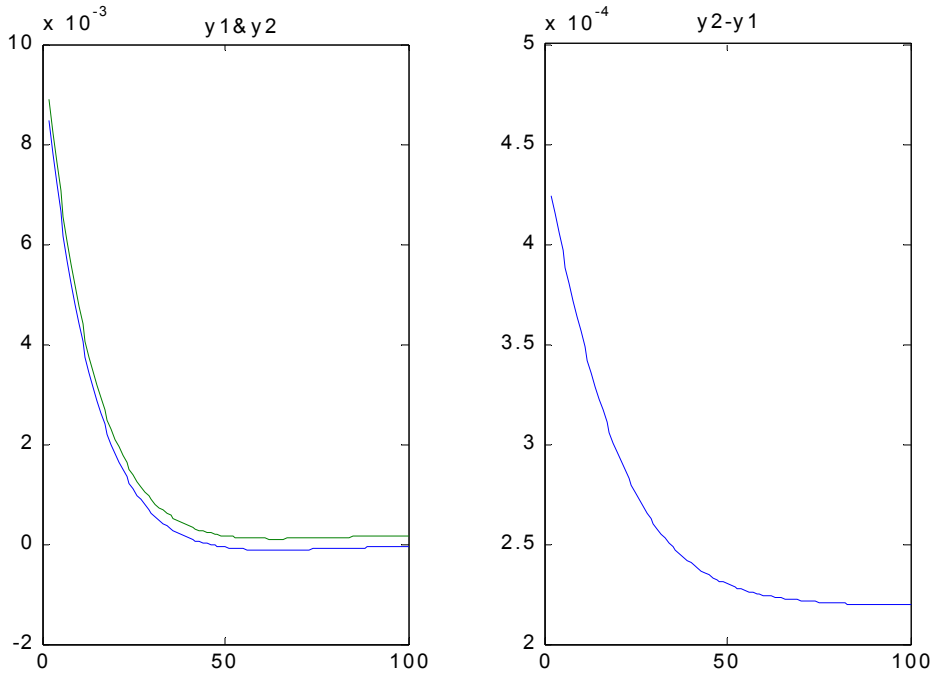

Figure16

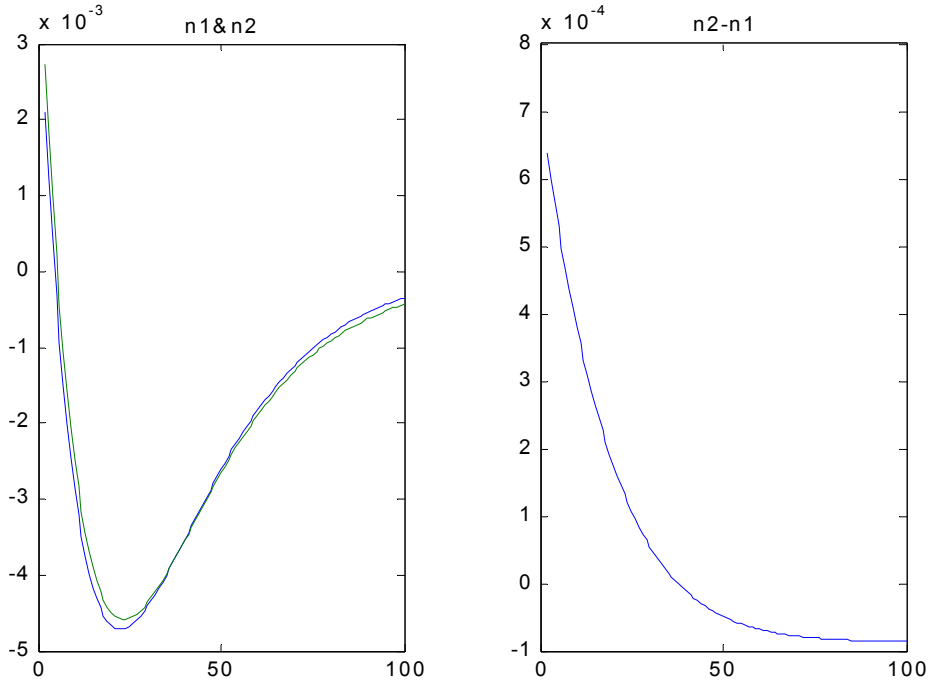

Figure17

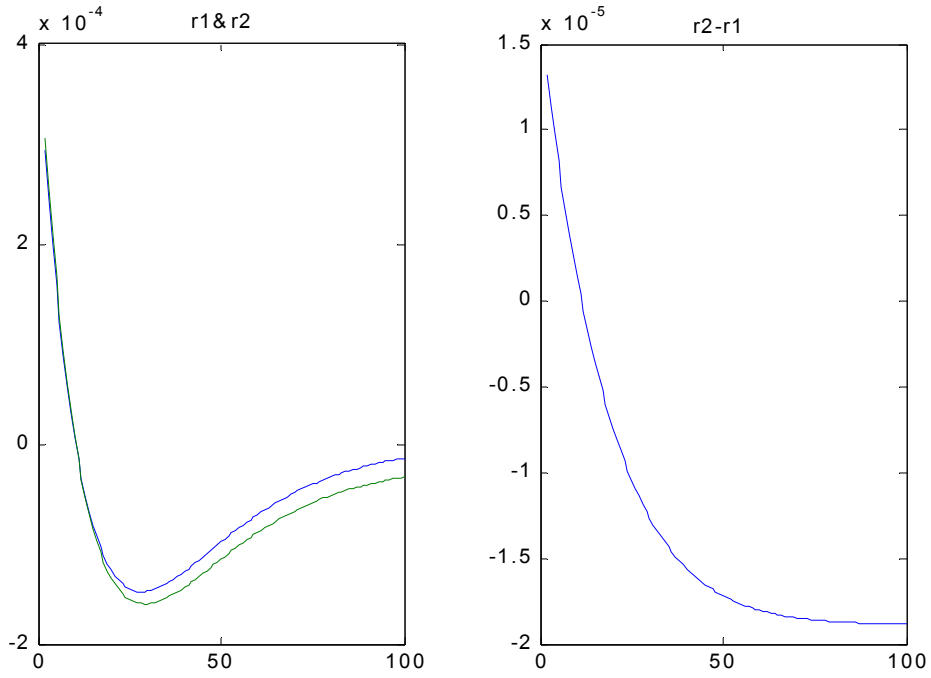

Table 2 ($\eta = 0.1$)

	First order		Second order	
	Standard Deviation	Correlation with output	Standard Deviation	Correlation with output
Technology	0.0216	0.8747	0.0218	0.8749
Capital	0.1586	0.9480	0.1602	0.9483
Consumption	0.1846	0.7853	0.1863	0.7870
Labor	0.1631	0.9976	0.1646	0.9976
Output	0.1774	1	0.1791	1
Interest	0.0019	0.4811	0.0019	0.4807

Table 3 ($\eta = 10$)

	First order		Second order	
	Standard Deviation	Correlation with output	Standard Deviation	Correlation with output
Technology	0.0217	0.9611	0.0217	0.9608
Capital	0.0282	0.4476	0.0282	0.4458
Consumption	0.0036	0.7653	0.0036	0.7642
Labor	0.0246	-0.2649	0.0246	-0.2623
Output	0.0210	1	0.0210	1
Interest	9.2000e-004	0.3170	9.2176e-004	0.3185

below the one at first order and the difference converges to about-0.0045. When $\eta = 10$, we see the difference of impulse responses between first order and second order is generally smaller than in the case of $\eta = 0.1$ but still larger than $\eta = 1$.

In Tables 2&3, we can see, like in the case of $\eta = 1$, the standard deviation and the correlation with the output of all variables are very close between first order and second order and some of the values are the same at our rounding level.

7. Discussion

Now we return to the two main questions we have asked at the beginning. The first one is how the higher approximation should be accurately conducted. As Schmitt-Grohe & Uribe (2002) state, spurious results arise in the common practice of evaluating a second order approximation to the objective function by using a first order approximation to the decision rule. In this case, some second order terms of the equilibrium welfare function are ignored while others not. And generally a correct second order approximation of the equilibrium welfare function requires a second order approximation to the policy function. So this example might be a justification for starting solving the model by second approximation to the policy function. In this paper, we apply this approach in a standard RBC model and find that the solution method works very well. It generates the same results at first order approximation as those by Uhlig's (1995) toolkit programs.

Moreover, when we compare the approach of approximation to the policy function with the usual approach, we find that when we apply these two approaches we choose the different starting points to approximate. In the case of first order, it has been shown that the two approaches lead to the same results. However, when we apply higher order approximation, the results might depend on what sort of equations we start to approximate and how we want to formulate the solutions.

The second question is what difference the second order approximation makes compared with first order approximation. In the examples mentioned above, we know in some cases, when we compare welfare across alternative policies, we will have to resort to second order approximation to the decision rule to avoid spurious results. And In this paper, we compare in some depth the numerical results of first order and second order approximations in a standard RBC model. In parts 5&6, we see that the numerical results from second order approximation are so often quite close to those from first order approximation, except that in second order approximation, the volatility enters the decision rules, which makes the second order decision rule different from the first order one. And we are always approximating the behaviors of the economic systems we build, and how well the approximation is done virtually depends on the question we ask. This paper did not build up specific scenarios to test what difference the second order approximation makes. Selecting some widely used models and their properties to test the second order approximation with regard to the first order approximation might be a further way to go.

8. Conclusion

In this paper, we have practiced the approach of second order approximation to the policy function in a standard RBC model. We have theoretically compared the approach of approximation to the policy function with the usual approach at the first order, and prove that they lead to the same system of equations to be solved for. And we have compared the impulse responses and calculated second moments from second order recursive law of motion with those from first order one. We find that the impulse response at second order converges to a new level due to the property of the second order recursive law of motion and the difference of impulse response between first order and second order is very small in the standard calibration but becomes larger for other values of relative risk aversion. The calculated second moments tend to be very close between first order and second order for all parameters tested.

Appendix

a. Proof

In this section, we briefly prove that at first order the two approaches lead to the same system of equations to be solved. For comparison convenience, we still use the Schmitt-Grohe & Uribe (2002) notation, which formulates the general set of equilibrium conditions of usual macroeconomic models as

$$E_t f(y', y, x', x) = 0\,, \tag{8}$$

First, by Schmitt-Grohe & Uribe (2002) approach, we have the general form of the policy functions and the first order approximated ones as follows:

$$y = g(x, \sigma)\,, \tag{10}$$

$$x' = h(x, \sigma) + \theta\sigma\varepsilon'\,, \tag{11}$$

$$g(x, \sigma) = g(\bar{x}, 0) + g_x(\bar{x}, 0)(x - \bar{x}) + g_\sigma(\bar{x}, 0)\sigma \tag{13}$$

$$h(x, \sigma) = h(\bar{x}, 0) + h_x(\bar{x}, 0)(x - \bar{x}) + h_\sigma(\bar{x}, 0)\sigma\,. \tag{14}$$

And we solve for those coefficients $g_x(\bar{x},0)$, $h_x(\bar{x},0)$, $g_\sigma(\bar{x},0)$ and $h_\sigma(\bar{x},0)$ from $F_x(\bar{x},0) = 0$ and $F_\sigma(\bar{x},0) = 0$ as follows:

$$\begin{aligned}
\left[F_x(\bar{x},0)\right]^i_j &= \left[f_{y'}\right]^i_\alpha \left[g_x\right]^\alpha_\beta \left[h_x\right]^\beta_j + \left[f_y\right]^i_\alpha \left[g_x\right]^\alpha_j + \left[f_{x'}\right]^i_\beta \left[h_x\right]^\beta_j + \left[f_x\right]^i_j \\
&= 0\,; \\
\left[F_\sigma(\bar{x},0)\right]^i &= E_t\left\{\left[f_{y'}\right]^i_\alpha \left[g_x\right]^\alpha_\beta \left[h_\sigma\right]^\beta + \left[f_{y'}\right]^i_\alpha \left[g_x\right]^\alpha_\beta \left[\theta\right]^\beta_\phi \left[\varepsilon'\right]^\phi + \left[f_{y'}\right]^i_\alpha \left[g_\sigma\right]^\alpha + \left[f_y\right]^i_\alpha \left[g_\sigma\right]^\alpha \right. \\
&\qquad \left. + \left[f_{x'}\right]^i_\beta \left[h_\sigma\right]^\beta + \left[f_{x'}\right]^i_\beta \left[\theta\right]^\beta_\phi \left[\varepsilon'\right]^\phi \right\} \\
&= \left[f_{y'}\right]^i_\alpha \left[g_x\right]^\alpha_\beta \left[h_\sigma\right]^\beta + \left[f_{y'}\right]^i_\alpha \left[g_\sigma\right]^\alpha + \left[f_y\right]^i_\alpha \left[g_\sigma\right]^\alpha + \left[f_{x'}\right]^i_\beta \left[h_x\right]^\beta \\
&= 0\,.
\end{aligned}$$

where $i = 1,...,n$; $j, \beta = 1,...,n_x$; $\alpha = 1,...,n_y$ and $\phi = 1,...,n_\varepsilon$.

Now, we employ the usual approach. We first linearize the whole system (8),

$$E_t([f_{y'}]^i_\alpha [dy']^\alpha + [f_y]^i_\alpha [dy]^\alpha + [f_{x'}]^i_\beta [dx']^\beta + [f_x]^i_\beta [dx]^\beta) = 0; \tag{31}$$

$i = 1,...,n;\ \alpha = 1,...,n_y;\ \beta = 1,...,n_x$,

where $dy' = y' - \bar{y}$, $dy = y - \bar{y}$, $dx' = x' - \bar{x}$ and $dx = x - \bar{x}$, and (31) is a system of n equations.

Then we assume the recursive law of motion as

$$dy = Pdx + Q\sigma; \tag{32}$$

$$dx' = Mdx + N\sigma + \eta\sigma\varepsilon'. \tag{33}$$

They are the same as the approximated policy function in the above approach and coefficient matrices P, M, Q, and N are respectively $g_x(\bar{x},0)$, $h_x(\bar{x},0)$, $g_\sigma(\bar{x},0)$ and $h_\sigma(\bar{x},0)$. Plugging equations (32) and (33) into (31), we have

$$E_t([f_{y'}]^i_\alpha [PMdx + PN\sigma + P\eta\sigma\varepsilon' + Q\sigma]^\alpha + [f_y]^i_\alpha [Pdx + Q\sigma]^\alpha$$
$$+ [f_{x'}]^i_\beta [Mdx + N\sigma + \eta\sigma\varepsilon']^\beta + [f_x]^i_\beta [dx]^\beta) = 0,$$

$$([f_{y'}]^i_\alpha [g_x]^\alpha_\beta [h_x]^\beta_j + [f_y]^i_\alpha [g_x]^\alpha_j + [f_{x'}]^i_\beta [h_x]^\beta_j + [f_x]^i_j)[dx]^j$$
$$+ ([f_{y'}]^i_\alpha [g_x]^\alpha_\beta [h_\sigma]^\beta + [f_{y'}]^i_\alpha [g_\sigma]^\alpha + [f_y]^i_\alpha [g_\sigma]^\alpha + [f_{x'}]^i_\beta [h_x]^\beta)[\sigma] = 0.$$

As the above equation has to be zero for any values of x and σ, the sum of the coefficients of each variables has to be zero. Then we have

$$[f_{y'}]^i_\alpha [g_x]^\alpha_\beta [h_x]^\beta_j + [f_y]^i_\alpha [g_x]^\alpha_j + [f_{x'}]^i_\beta [h_x]^\beta_j + [f_x]^i_j = 0,$$

$$\left[f_{y'}\right]_{\alpha}^{i}\left[g_{x}\right]_{\beta}^{\alpha}\left[h_{\sigma}\right]^{\beta}+\left[f_{y'}\right]_{\alpha}^{i}\left[g_{\sigma}\right]^{\alpha}+\left[f_{y}\right]_{\alpha}^{i}\left[g_{\sigma}\right]^{\alpha}+\left[f_{x'}\right]_{\beta}^{i}\left[h_{x}\right]^{\beta}=0\,.$$

It is the same system of equations as the one to be solved for unknown coefficients in the approximated policy function.

b. Matlab Codes

1. RBC_model.m

```
function
[nfx,nfxp,nfy,nfyp,nfypyp,nfypy,nfypxp,nfypx,nfyyp,nfyy,nfyxp,nfyx,nfxpyp,nfxpy,nfx
pxp,nfxpx,nfxyp,nfxy,nfxxp,nfxx] = RBC_model
%This program computes numerical first and second derivatives of the  function f for the simple neoclassical growth model described in section 2.1 of ``Solving Dynamic General Equilibrium Models Using a Second-Order Approximation to the Policy Function," by Stephanie Schmitt-Grohe and Martin Uribe, (2001). Unlike the example in section 2.1, here y and x are defined as log(c) and [log(k); log(A)] respectively.  The function f defines  the DSGE model:
%  E_t f(yp,y,xp,x) =0.
%
%Inputs: none
%
%Output: Numerical first and second derivatives of f
%
%Calls: anal_deriv.m num_eval.m RBC_model_ss.m
%
%(c) Stephanie Schmitt-Grohe and Martin Uribe
%Date July 17, 2001

%Define parameters
syms    SIG DELLTA ALFA BETTA RHO A

%Define variables
syms c cp k kp a ap

%Write equations ei, i=1:3
e1 = exp(c) + exp(kp) - (1-DELLTA) * exp(k) - exp(a) * exp(k)^ALFA*((A/(exp(c)^(-
```

```
SIG)*(1-ALFA)*exp(a)*exp(k)^ALFA))^(-1/ALFA))^(1-ALFA);
e2 = exp(c)^(-SIG) - BETTA * exp(cp)^(-SIG) * (exp(ap) * ALFA *((A/(exp(cp)^(-
SIG)*(1-ALFA)*exp(ap)*exp(kp)^ALFA))^(-1/ALFA))^(1-ALFA)* exp(kp)^(ALFA-1)
+ 1 - DELLTA);
e3 = ap - RHO * a;

%Create function f
f = [e1;e2;e3];

% Define the vector of controls, y, and states, x
x = [k a];
y = [c];
xp = [kp ap];
yp = [cp];
nx = length(x);
ny = length(y);

%Compute analytical derivatives of f
[fx,fxp,fy,fyp,fypyp,fypy,fypxp,fypx,fyyp,fyy,fyxp,fyx,fxpyp,fxpy,fxpxp,fxpx,fxyp,fxy,f
xxp,fxx]=anal_deriv(f,x,y,xp,yp);

%Numerical Evaluation
%Steady State and Parameter Values
[SIG,DELLTA,ALFA,BETTA,RHO,A,c,cp,k,kp,a,ap,]=RBC_model_ss;
%Order of approximation desired
approx = 2;
%Obtain numerical derivatives of f
num_eval
```

2. RBC_model_ss.m

```
function
[nSIGMA,nDELTA,nALFA,nBETTA,nRHO,nA,nc,ncp,nk,nkp,na,nap,nn,nnp]=RBC_model_ss
%This program produces the the deep structural parameters and computes the steady state of the simple neoclassical growth model described in section 2.1 of ``Solving Dynamic General Equilibrium Models Using a Second-Order Approximation to the Policy Function," by Stephanie Schmitt-Grohe and Martin Uribe, (2001).
%
%(c) Stephanie Schmitt-Grohe and Martin Uribe
%Date July 17, 2001

nBETTA=1/1.01;      %discount rate
nDELTA=0.025;       %depreciation rate
nALFA=0.36;         %capital share
nRHO=0.95;          %persistence of technology shock
nSIGMA=10;          %intertemporal elasticity of substitution

n = 1/3;            %labor share
a = 1;              %steady-state value of technology shock
k  =  ((1/nBETTA+nDELTA-1)/(nALFA*n^(1-nALFA)))^(1/(nALFA-1));        %steady-state value of capital
c = a * k^(nALFA)*n^(1-nALFA)-nDELTA*k;                    %steady-state value of consumption
nA= c^(-nSIGMA)*(1-nALFA)*k^(nALFA)*n^(-nALFA);                %labor parameter

na = log(a);
nk = log(k);
nc = log(c);
nn = log(n);
```

```
nap=na;
nkp=nk;
ncp=nc;
nnp=nn;
```

3. RBC_model_run.m

```
%This program computes a second-order  approximation to the policy functions of a simple neoclassical model (see ``Solving Dynamic General Equilibrium Models Using a Second-Order Approximation to the Policy Function," by Stephanie Schmitt-Grohe and Martin Uribe, (2001). The reduced form of the model can be written as:
%E_t[f(yp,y,xp,x)=0,
%The solution is of the form
%xp = h(x,sigma) + sigma* eta * ep
%y = g(x,sigma)
%The quadratic approximation to these functions are (in tensor notation) [Notation: x is x_t and xp is x_t+1, variables are expressed in log-deviations from their steady state value]
%xp^i = hx^i_a x_a + 1/2 [hxx^i_ab x_a x_b + hss^i sigma^2] + sigma*  eta^i_c ep_c
%y^i = gx^i_a x_a + 1/2 [gxx^i_ab x_a x_b + gss^i sigma^2]
%
%where
% hx is nx by nx
% gx is ny by nx
% hxx is nx by nx by nx
% gxx is ny by nx by nx
% eta is nx by ne
% gss is ny by 1
% hss is nx by 1
% sigma is a positive scalar
```

```
%Calls: neoclassical_model.m gx_hx.m gxx_hxx.m gss_hss.m
%
%(c) Stephanie Schmitt-Grohe and Martin Uribe
%Date July 17, 2001

[fx,fxp,fy,fyp,fypyp,fypy,fypxp,fypx,fyyp,fyy,fyxp,fyx,fxpyp,fxpy,fxpxp,fxpx,fxyp,fxy,f
xxp,fxx]=RBC_model;

[gx,hx] = gx_hx(fy,fx,fyp,fxp)

[gxx,hxx]                                                                    =
gxx_hxx(fx,fxp,fy,fyp,fypyp,fypy,fypxp,fypx,fyyp,fyy,fyxp,fyx,fxpyp,fxpy,fxpxp,fxpx,f
xyp,fxy,fxxp,fxx,hx,gx)

eta=[0 1]';

[gss,hss]                                                                    =
gss_hss(fx,fxp,fy,fyp,fypyp,fypy,fypxp,fypx,fyyp,fyy,fyxp,fyx,fxpyp,fxpy,fxpxp,fxpx,fx
yp,fxy,fxxp,fxx,hx,gx,gxx,eta)
```

4. RBC_impulse

```
%calibration and steady states

N_bar    = 1.0/3;     % Steady state employment is a third of total time endowment
Z_bar    = 1;         % Normalization
rho      = .36;       % Capital share
delta    = .025;      % Depreciation rate for capital
R_bar    = 1.01;      % One percent real interest per quarter
eta      = 10         % constant of relative risk aversion = 1/(coeff. of intertemporal
```

```
substitution)
psi      = .95;       % autocorrelation of technology shock

% Calculating the steady state:

betta   = 1.0/R_bar;                          % Discount factor beta
YK_bar  = (R_bar + delta - 1)/rho;                    % = Y_bar / K_bar
K_bar   = (YK_bar / Z_bar)^(1.0/(rho-1)) * N_bar;
I_bar   = delta * K_bar;
Y_bar   = YK_bar * K_bar;
C_bar   = Y_bar - delta*K_bar;
A       =  C_bar^(-eta) * (1 - rho) * Y_bar/N_bar;      % Parameter in utility function

%impulse response

T=100;                %time length

%at first order

z1=zeros(T,1);
k1=zeros(T,1);
c1=zeros(T,1);
y1=zeros(T,1);
n1=zeros(T,1);
r1=zeros(T,1);
esp=zeros(T,1);

esp(2,1)=1;               %
sigma_eps=0.00712;                %scaling parameter
```

```
for t=2:T
z1(t)=hx(2,2)*z1(t-1)+sigma_eps*esp(t);
k1(t)=hx(1,1)*k1(t-1)+hx(1,2)*z1(t-1);
c1(t)=gx(1,1)*k1(t)+gx(1,2)*z1(t);
n1(t)=-eta/rho*c1(t)+1/rho*z1(t)+k1(t);
y1(t)=(1-rho)*n1(t)+z1(t)+rho*k1(t);
r1(t)=rho*Y_bar/(K_bar*R_bar)*(y1(t)-k1(t));
end


% at second order

z2=zeros(T,1);
k2=zeros(T,1);
c2=zeros(T,1);
y2=zeros(T,1);
n2=zeros(T,1);
i2=zeros(T,1);
r2=zeros(T,1);


for t=2:T
z2(t)=hx(2,2)*z2(t-1)+sigma_eps*esp(t);
k2(t)=hx(1,1)*k2(t-1)+hx(1,2)*z2(t-1)+.5*hxx(1,1,1)*k2(t-1)^2+hxx(1,2,1)*k2(t-
1)*z2(t-1)+.5*hxx(2,2,2)*z2(t-1)^2+.5*hss(1,1)*sigma_eps^2;
c2(t)=gx(1,1)*k2(t)+gx(1,2)*z2(t)+.5*gxx(1,1,1)*k2(t)^2+gxx(1,2,1)*k2(t)*z2(t)+.5*g
xx(1,2,2)*z2(t)^2+.5*gss*sigma_eps^2;
n2(t)=-eta/rho*c2(t)+1/rho*z2(t)+k2(t);
y2(t)=(1-rho)*n2(t)+z2(t)+rho*k2(t);
r2(t)=rho*Y_bar/(K_bar*R_bar)*(y2(t)-k2(t));
```

```
end


%difference of series between first order and second order

difk=k2-k1;
difc=c2-c1;
difn=n2-n1;
dify=y2-y1;
difr=r2-r1;


%plot impulse response in one graph at first order
plot(2:T,z1(2:T),2:T,k1(2:T),2:T,c1(2:T),2:T,y1(2:T),2:T,n1(2:T),2:T,r1(2:T));
figure;

%plot impulse response in one graph at second order
plot(2:T,z2(2:T),2:T,k2(2:T),2:T,c2(2:T),2:T,y2(2:T),2:T,n2(2:T),2:T,r2(2:T));
figure;

%plot techonology
plot(2:T,z1(2:T));
figure;

%plot k1 & k2 & their difference
subplot(1,2,1);
plot(2:T,k1(2:T),2:T,k2(2:T));

subplot(1,2,2);
plot(2:T,difk(2:T));
```

```
figure;

%plot c1 & c2 & their difference
subplot(1,2,1);
plot(2:T,c1(2:T),2:T,c2(2:T));

subplot(1,2,2);
plot(2:T,difc(2:T));
figure;

%plot y1 & y2 & their difference
subplot(1,2,1);
plot(2:T,y1(2:T),2:T,y2(2:T));

subplot(1,2,2);
plot(2:T,dify(2:T));
figure;

%plot n1 & n2 & their difference
subplot(1,2,1);
plot(2:T,n1(2:T),2:T,n2(2:T));

subplot(1,2,2);
plot(2:T,difn(2:T));
figure;

%plot r1 & r2 & their difference
subplot(1,2,1);
plot(2:T,r1(2:T),2:T,r2(2:T));
```

```
subplot(1,2,2);
plot(2:T,difr(2:T));
figure;
```

5. RBC_model_simulation

```
% Calculating the steady state:
Z_bar    = 1;    % Normalization
rho      = .36;  % Capital share
delta    = .025; % Depreciation rate for capital
R_bar    = 1.01; % One percent real interest per quarter
eta          = 10;      % constant of relative risk aversion = 1/(coeff. of intertemporal substitution)
psi      = .95;  % autocorrelation of technology shock
N_bar    = 1.0/3;      % Steady state employment is a third of total time endowment

betta   = 1.0/R_bar;                          % Discount factor beta
YK_bar  = (R_bar + delta - 1)/rho;                    % = Y_bar / K_bar
K_bar   = (YK_bar / Z_bar)^(1.0/(rho-1)) * N_bar;
I_bar   = delta * K_bar;
Y_bar   = YK_bar * K_bar;
C_bar   = Y_bar - delta*K_bar;
A       = C_bar^(-eta) * (1 - rho) * Y_bar/N_bar;      % Parameter in utility function

%simulation parameters

sigma_eps=0.00712;    %scaling parameter
T=600;              %time length
U=1000;               %times of simulation
```

```
%simulation of first order approximation

z1=zeros(T,1);
k1=zeros(T,1);
c1=zeros(T,1);
y1=zeros(T,1);
n1=zeros(T,1);
i1=zeros(T,1);
r1=zeros(T,1);
simuz1=zeros(U,T);
simuk1=zeros(U,T);
simuc1=zeros(U,T);
simuy1=zeros(U,T);
simun1=zeros(U,T);
simui1=zeros(U,T);
simur1=zeros(U,T);


for s1=1:U
esp=randn(T,1);
for t=2:T
z1(t)=hx(2,2)*z1(t-1)+sigma_eps*esp(t);
k1(t)=hx(1,1)*k1(t-1)+hx(1,2)*z1(t-1);
c1(t)=gx(1,1)*k1(t)+gx(1,2)*z1(t);
n1(t)=-eta/rho*c1(t)+1/rho*z1(t)+k1(t);
y1(t)=(1-rho)*n1(t)+z1(t)+rho*k1(t);
r1(t)=rho*Y_bar/(K_bar*R_bar)*(y1(t)-k1(t));
end
simuz1(s1,:)=z1';
simuk1(s1,:)=k1';
```

```
simuc1(s1,:)=c1';
simun1(s1,:)=n1';
simuy1(s1,:)=y1';
simur1(s1,:)=r1';
end


%calculation of std deviation

stdz1=mean(std(simuz1'))
stdk1=mean(std(simuk1'))
stdc1=mean(std(simuc1'))
stdn1=mean(std(simun1'))
stdy1=mean(std(simuy1'))
stdr1=mean(std(simur1'))

%calculation of correlation
corr_zy1=zeros(U,1);
corr_ky1=zeros(U,1);
corr_cy1=zeros(U,1);
corr_yy1=zeros(U,1);
corr_ny1=zeros(U,1);
corr_ry1=zeros(U,1);

for j1=1:U
AA=corrcoef(simuz1(j1,:),simuy1(j1,:));
corr_zy1(j1)=AA(1,2);
BB=corrcoef(simuk1(j1,:),simuy1(j1,:));
corr_ky1(j1)=BB(1,2);
CC=corrcoef(simuc1(j1,:),simuy1(j1,:));
```

```
corr_cy1(j1)=CC(1,2);
DD=corrcoef(simuy1(j1,:),simuy1(j1,:));
corr_yy1(j1)=DD(1,2);
EE=corrcoef(simun1(j1,:),simuy1(j1,:));
corr_ny1(j1)=EE(1,2);
FF=corrcoef(simur1(j1,:),simuy1(j1,:));
corr_ry1(j1)=FF(1,2);
end

cor_zy1=mean(corr_zy1)
cor_ky1=mean(corr_ky1)
cor_cy1=mean(corr_cy1)
cor_yy1=mean(corr_yy1)
cor_ny1=mean(corr_ny1)
cor_ry1=mean(corr_ry1)


%simulation of second order approximation


z2=zeros(T,1);
k2=zeros(T,1);
c2=zeros(T,1);
y2=zeros(T,1);
n2=zeros(T,1);
r2=zeros(T,1);
simuz2=zeros(U,T);
simuk2=zeros(U,T);
simuc2=zeros(U,T);
simuy2=zeros(U,T);
```

```
simun2=zeros(U,T);
simur2=zeros(U,T);

for s2=1:U
esp=randn(T,1);
for t=2:T
z2(t)=hx(2,2)*z2(t-1)+sigma_eps*esp(t);
k2(t)=hx(1,1)*k2(t-1)+hx(1,2)*z2(t-1)+.5*hxx(1,1,1)*k2(t-1)^2+hxx(1,2,1)*k2(t-
1)*z2(t-1)+.5*hxx(2,2,2)*z2(t-1)^2+.5*hss(1,1)*sigma_eps^2;
c2(t)=gx(1,1)*k2(t)+gx(1,2)*z2(t)+.5*gxx(1,1,1)*k2(t)^2+gxx(1,2,1)*k2(t)*z2(t)+.5*g
xx(1,2,2)*z2(t)^2+.5*gss*sigma_eps^2;
n2(t)=-eta/rho*c2(t)+1/rho*z2(t)+k2(t);
y2(t)=(1-rho)*n2(t)+z2(t)+rho*k2(t);
r2(t)=rho*Y_bar/(K_bar*R_bar)*(y2(t)-k2(t));
end
simuz2(s2,:)=z2';
simuk2(s2,:)=k2';
simuc2(s2,:)=c2';
simun2(s2,:)=n2';
simuy2(s2,:)=y2';
simur2(s2,:)=r2';
end

%std deviation

stdz2=mean(std(simuz2'))
stdk2=mean(std(simuk2'))
stdc2=mean(std(simuc2'))
stdn2=mean(std(simun2'))
stdy2=mean(std(simuy2'))
```

```
stdr2=mean(std(simur2'))


corr_zy2=zeros(U,1);
corr_ky2=zeros(U,1);
corr_cy2=zeros(U,1);
corr_yy2=zeros(U,1);
corr_ny2=zeros(U,1);
corr_ry2=zeros(U,1);

for j2=1:U
AA=corrcoef(simuz2(j2,:),simuy2(j2,:));
corr_zy2(j2)=AA(1,2);
BB=corrcoef(simuk2(j2,:),simuy2(j2,:));
corr_ky2(j2)=BB(1,2);
CC=corrcoef(simuc2(j2,:),simuy2(j2,:));
corr_cy2(j2)=CC(1,2);
DD=corrcoef(simuy2(j2,:),simuy2(j2,:));
corr_yy2(j2)=DD(1,2);
EE=corrcoef(simun2(j2,:),simuy2(j2,:));
corr_ny2(j2)=EE(1,2);
FF=corrcoef(simur2(j2,:),simuy2(j2,:));
corr_ry2(j2)=FF(1,2);
end

cor_zy2=mean(corr_zy2)
cor_ky2=mean(corr_ky2)
cor_cy2=mean(corr_cy2)
cor_yy2=mean(corr_yy2)
cor_ny2=mean(corr_ny2)
```

```
cor_ry2=mean(corr_ry2)
```

6. anal_deriv.m

```
function
[fx,fxp,fy,fyp,fypyp,fypy,fypxp,fypx,fyyp,fyy,fyxp,fyx,fxpyp,fxpy,fxpxp,fxpx,fxyp,fxy,f
xxp,fxx]=anal_deriv(f,x,y,xp,yp,approx);
% This program copmutes analytical first and second (if approx=2) derivatives of the
function f(yp,y,xp,x) with respect to x, y, xp, and yp.  For documentation, see the paper
``Solving Dynamic General Equilibrium Models Using a Second-Order Approximation
to the Policy Function," by Stephanie Schmitt-Grohe and Martin Uribe, 2001).
%
%Inputs: f, x, y, xp, yp, approx
%
%Output: Analytical first and second derivatives of f.
%
%If approx is set at a value different from 2, the program delivers the first derivatives of
f and sets second derivatives at zero. If approx equals 2, the program returns first and
second derivatives of f. The default value of approx is 2.
%Note: This program requires MATLAB's Symbolic Math Toolbox
%
%(c) Stephanie Schmitt-Grohe and Martin Uribe
%Date July 17, 2001


if nargin==5
approx=2;
end

nx = size(x,2);
```

```
ny = size(y,2);
nxp = size(xp,2);
nyp = size(yp,2);

n = size(f,1);

%Compute the first and second derivatives of f
fx = jacobian(f,x);
fxp = jacobian(f,xp);
fy = jacobian(f,y);
fyp = jacobian(f,yp);

if approx==2

fypyp = reshape(jacobian(fyp(:),yp),n,nyp,nyp);

fypy = reshape(jacobian(fyp(:),y),n,nyp,ny);

fypxp = reshape(jacobian(fyp(:),xp),n,nyp,nxp);

fypx = reshape(jacobian(fyp(:),x),n,nyp,nx);

fyyp = reshape(jacobian(fy(:),yp),n,ny,nyp);

fyy = reshape(jacobian(fy(:),y),n,ny,ny);

fyxp = reshape(jacobian(fy(:),xp),n,ny,nxp);

fyx = reshape(jacobian(fy(:),x),n,ny,nx);
```

```
fxpyp = reshape(jacobian(fxp(:),yp),n,nxp,nyp);

fxpy = reshape(jacobian(fxp(:),y),n,nxp,ny);

fxpxp = reshape(jacobian(fxp(:),xp),n,nxp,nxp);

fxpx = reshape(jacobian(fxp(:),x),n,nxp,nx);

fxyp = reshape(jacobian(fx(:),yp),n,nx,nyp);

fxy = reshape(jacobian(fx(:),y),n,nx,ny);

fxxp = reshape(jacobian(fx(:),xp),n,nx,nxp);

fxx = reshape(jacobian(fx(:),x),n,nx,nx);

else

fypyp=0; fypy=0; fypxp=0; fypx=0; fyyp=0; fyy=0; fyxp=0; fyx=0; fxpyp=0; fxpy=0;
fxpxp=0; fxpx=0; fxyp=0; fxy=0; fxxp=0; fxx=0;

end
```

7. num_eval.m

```
%This program evaluates the analytical first and second (if approx=2) derivatives of f
numerically. The parameters and steady state values of the arguments of the function f
are assumed to be in the workspace. Also, the order of approximation must be in the
workspace.
%
```

```
%(c) Stephanie Schmitt-Grohe and Martin Uribe
%Date July 17, 2001
%Changed on September 25, 2001 to replace subs with eval and make it no longer a
function.


nfx = zeros(size(fx));
nfx(:) = eval(fx(:));

nfxp = zeros(size(fxp));
nfxp(:)= eval(fxp(:));

nfy = zeros(size(fy));
nfy(:) = eval(fy(:));

nfyp = zeros(size(fyp));
nfyp(:)= eval(fyp(:));

nf = zeros(size(f));
nf(:)=eval(f(:));

if approx==1

%If only a first-order approximation is desired, set all second derivatives equal to zero
nfypyp=0; nfypy=0; nfypxp=0; nfypx=0; nfyyp=0; nfyy=0; nfyxp=0; nfyx=0; nfxpyp=0;
nfxpy=0; nfxpxp=0; nfxpx=0; nfxyp=0; nfxy=0; nfxxp=0; nfxx=0;

else

nfypyp=zeros(size(fypyp));
```

```
nfypyp(:)=eval(fypyp(:));

nfypy=zeros(size(fypy));
nfypy(:)=eval(fypy(:));

nfypxp=zeros(size(fypxp));
nfypxp(:)=eval(fypxp(:));

nfypx=zeros(size(fypx));
nfypx(:)=eval(fypx(:));

nfyyp=zeros(size(fyyp));
nfyyp(:)=eval(fyyp(:));

nfyy=zeros(size(fyy));
nfyy(:)=eval(fyy(:));

nfyxp=zeros(size(fyxp));
nfyxp(:)=eval(fyxp(:));

nfyx=zeros(size(fyx));
nfyx(:)=eval(fyx(:));

nfxpyp=zeros(size(fxpyp));
nfxpyp(:)=eval(fxpyp(:));

nfxpy=zeros(size(fxpy));
nfxpy(:)=eval(fxpy(:));

nfxpxp=zeros(size(fxpxp));
```

```
nfxpxp(:)=eval(fxpxp(:));

nfxpx=zeros(size(fxpx));
nfxpx(:)=eval(fxpx(:));

nfxyp=zeros(size(fxyp));
nfxyp(:)=eval(fxyp(:));

nfxy=zeros(size(fxy));
nfxy(:)=eval(fxy(:));

nfxxp=zeros(size(fxxp));
nfxxp(:)=eval(fxxp(:));

nfxx=zeros(size(fxx));
nfxx(:)=eval(fxx(:));



end
```

8. gx_hx.m

```
function [gx,hx] = gx_hx(fy,fx,fyp,fxp);
%This program computes the matrices gx and hx that define the first-order approximation
%of the DSGE model. That is, if
%E_t[f(yp,y,xp,x)=0, then the solution is of the form
%xp = h(x,sigma) + sigma * eta * ep
%y = g(x,sigma).
%The first-order approximations to the functions g and h around the point (x,sigma)=(xbar,0), where xbar=h(xbar,0), are:
```

```
%h(x,sigma) = xbar + hx (x-xbar)
%and
%g(x,sigma) = ybar + gx * (x-xbar),
%where ybar=g(xbar,0).
%Inputs: fy fyp fx fxp
%Outputs: gx hx
%Calls solab.m (by Paul Klein)
%(c) Stephanie Schmitt-Grohe and Martin Uribe
%Date July 17, 2001
%old version
A = [-fxp -fyp];
B = [fx fy ];
%newly changed!!
%A = [-fxp];
%B = [fx];

[gx,hx]=solab(A,B,size(fx,2));
```

9. gxx_hxx.m

```
function [gxx,hxx] = gxx_hxx(fx,fxp,fy,fyp,fypyp,fypy,fypxp,fypx,fyyp,fyy,fyxp,fyx,fxpyp,fxpy,fxpxp,fxpx,fxyp,fxy,fxxp,fxx,hx,gx)

%This program finds the 3-dimensional arrays gxx and hxx necessary to compute the 2nd order approximation to the decision rules of a DSGE model of the form E_tf(yp,y,xp,x)=0, with solution xp=h(x,sigma) + sigma * eta * ep and y=g(x,sigma). For documentation, see the paper ``Solving Dynamic General Equilibrium Models Using a Second-Order Approximation to the Policy Function," by Stephanie Schmitt-Grohe and Martin Uribe, 2001)
```

%INPUTS: First and second derivatives of f and first-order approximation to the functions g and h: fx,fxp,fy,fyp,fypyp,fypy,fypxp,fypx,fyyp,fyy,fyxp,fyx,fxpyp,fxpy,fxpxp,fxpx,fxyp,fxy,fxxp,fxx,hx,gx

%OUTPUTS: Second-order derivatives of the functions g and h with respect to x, evaluated at (x,sigma)=(xbar,0), where xbar=h(xbar,0). That is, hxx gxx

% We solve a linear system of the type q = Q * x where x is a vector containing the elements of gxx and hxx appropritely stacked and q and Q are, respectively, a vector and a matrix whose elements are functions of the inputs of the program.

%(c) Stephanie Schmitt-Grohe and Martin Uribe

%Date July 17, 2001

```
m=0;
nx = size(hx,1); %rows of hx and hxx
ny = size(gx,1); %rows of gx and gxx
n = nx + ny; %length of f
ngxx = nx^2*ny; %elements of gxx

sg = [ny nx nx]; %size of gxx
sh = [nx nx nx]; %size of hxx

Q = zeros(n*nx*nx,n*nx*nx);
gxx=zeros(sg);
hxx=zeros(sh);
GXX=zeros(sg);
```

```
HXX=zeros(sh);

for i=1:n
for j=1:nx
for k=1:nx

m = m+1;

%First Term
q(m,1) = ( shiftdim(fypyp(i,:,:),1) * gx * hx(:,k) + shiftdim(fypy(i,:,:),1) * gx(:,k) +
shiftdim(fypxp(i,:,:),1) *  hx(:,k) + shiftdim(fypx(i,:,k),1) )' * gx * hx(:,j);

% Second term

GXX(:) = kron(ones(nx^2,1),fyp(i,:)');

pGXX = permute(GXX,[2 3 1]);
pGXX(:) = pGXX(:) .* kron(ones(nx*ny,1),hx(:,j));
GXX=ipermute(pGXX,[2 3 1]);

pGXX = permute(GXX,[3 1 2]);
pGXX(:) = pGXX(:) .* kron(ones(nx*ny,1),hx(:,k));
GXX=ipermute(pGXX,[3 1 2]);

Q(m,1:ngxx)=GXX(:)';

GXX=0*GXX;

%Third term
```

```
HXX(:,j,k) = (fyp(i,:) * gx)';

Q(m,ngxx+1:end)=HXX(:)';

HXX = 0*HXX;

%Fourth Term
q(m,1) = q(m,1) + ( shiftdim(fyyp(i,:,:),1) * gx * hx(:,k) +  shiftdim(fyy(i,:,:),1) * gx(:,k)
+ shiftdim(fyxp(i,:,:),1) * hx(:,k) +  shiftdim(fyx(i,:,k),1) )' * gx(:,j);

%Fifth Term

GXX(:,j,k)=fy(i,:)';

Q(m,1:ngxx) = Q(m,1:ngxx) + GXX(:)';

GXX = 0*GXX;

%Sixth term
q(m,1) = q(m,1) + ( shiftdim(fxpyp(i,:,:),1) * gx * hx(:,k) + shiftdim(fxpy(i,:,:),1) *
gx(:,k) + shiftdim(fxpxp(i,:,:),1) * hx(:,k) + fxpx(i,:,k)')' * hx(:,j);


%Seventh Term

HXX(:,j,k)=fxp(i,:)';

Q(m,ngxx+1:end) = Q(m,ngxx+1:end) + HXX(:)';

HXX = 0*HXX;
```

```
%Eighth Term
q(m,1) = q(m,1) +  shiftdim(fxyp(i,j,:),1) * gx * hx(:,k) +  shiftdim(fxy(i,j,:),1) * gx(:,k)
+  shiftdim(fxxp(i,j,:),1) * hx(:,k) + fxx(i,j,k);

end %k
end %j
end %i

x=-inv(Q)*q;

gxx(:)=x(1:ngxx);
hxx(:) = x(ngxx+1:end);
```

10. gss_hss.m

```
function [gss,hss] = gss_hss(fx,fxp,fy,fyp,fypyp,fypy,fypxp,fypx,fyyp,fyy,fyxp,fyx,fxpyp,fxpy,fxpxp,fxpx,fxyp,fxy,fxxp,fxx,hx,gx,gxx,eta)
%Finds the vectors gss and hss necessary to compute the 2nd order approximation to the decision rules of a DSGE model. For documentation, see the paper ``Solving Dynamic General Equilibrium Models Using a Second-Order Approximation to the Policy Function," by Stephanie Schmitt-Grohe and Martin Uribe, 2001)

%INPUTS:
fx,fxp,fy,fyp,fypyp,fypy,fypxp,fypx,fyyp,fyy,fyxp,fyx,fxpyp,fxpy,fxpxp,fxpx,fxyp,fxy,fxxp,fxx,hx,gx,gxx,eta%OUTPUTS: hss gss

% We solve a linear system of the type q = Q * x where x=[gss; hss];
```

```
%(c) Stephanie Schmitt-Grohe and Martin Uribe
%Date July 17, 2001

nx = size(hx,1); %rows of hx and hss
ny = size(gx,1); %rows of gx and gss
n = nx + ny;
ne = size(eta,2); %number of exogenous shocks (columns of eta)

for i=1:n

%First Term
Qh(i,:) = fyp(i,:) * gx;

%Second Term
q(i,1) = sum( diag( ( shiftdim(fypyp(i,:,:),1) * gx * eta)' * gx * eta ));

%Third Term
q(i,1) = q(i,1) + sum( diag(( shiftdim(fypxp(i,:,:),1) *  eta)' * gx * eta ));



%Fourth Term
fyp(i,:) * reshape(gxx,ny,nx^2);

q(i,1) =  q(i,1) + sum( diag(( reshape(ans,nx,nx) * eta )' * eta ));

%Fifth Term
Qg(i,:) = fyp(i,:);

%Sixth Term
Qg(i,:) = Qg(i,:) + fy(i,:);
```

```
%Seventh Term
Qh(i,:) = Qh(i,:) + fxp(i,:);

%Eighth Term
q(i,1) = q(i,1) + sum( diag( ( shiftdim(fxpyp(i,:,:),1) * gx * eta)' * eta ));

%Nineth Term
q(i,1) = q(i,1) + sum(diag( ( shiftdim(fxpxp(i,:,:),1) * eta)' * eta ));



end %i

x=-inv([Qg Qh])*q;

gss=x(1:ny);

hss = x(ny+1:end);
```

11.qzswitch.m

```
function [A,B,Q,Z] = qzswitch(i,A,B,Q,Z)
%function [A,B,Q,Z] = qzswitch(i,A,B,Q,Z)
% Written by Chris Sims
% Takes U.T. matrices A, B, orthonormal matrices Q,Z, interchanges
% diagonal elements i and i+1 of both A and B, while maintaining
% Q'AZ' and Q'BZ' unchanged.  Does nothing if ratios of diagonal elements
% in A and B at i and i+1 are the same.  Aborts if diagonal elements of
% both A and B are zero at either position.
%
```

```
a = A(i,i); d = B(i,i); b = A(i,i+1); e = B(i,i+1);
c = A(i+1,i+1); f = B(i+1,i+1);
wz = [c*e-f*b, (c*d-f*a)'];
xy = [(b*d-e*a)', (c*d-f*a)'];
n = sqrt(wz*wz');
m = sqrt(xy*xy');
if n == 0
return
else
wz = n\wz;
xy = m\xy;
wz = [wz; -wz(2)', wz(1)'];
xy = [xy;-xy(2)', xy(1)'];
A(i:i+1,:) = xy*A(i:i+1,:);
B(i:i+1,:) = xy*B(i:i+1,:);
A(:,i:i+1) = A(:,i:i+1)*wz;
B(:,i:i+1) = B(:,i:i+1)*wz;
Z(:,i:i+1) = Z(:,i:i+1)*wz;
Q(i:i+1,:) = xy*Q(i:i+1,:);
End
```

12. solab.m

```
%
% Function: solab
%
% Purpose: Solves for the recursive representation of the stable solution to a system
% of linear difference equations.
%
% Inputs: Two square matrices a and b and a natural number nk
%
```

```
% a and b are the coefficient matrices of the difference equation
%
% a*x(t+1) = b*x(t)
%
% where x(t) is arranged so that the state variables come first, and
%
% nk is the number of state variables.
%
% Outputs: the decision rule f and the law of motion p. If we write
%
% x(t) = [k(t);u(t)] where k(t) contains precisely the state variables, then
%
% u(t)   = f*k(t) and
%
% k(t+1) = p*k(t).
%
% Calls: reorder
%

function [f,p] = solab(a,b,nk);

[s,t,q,z] = qz(a,b);            % upper triangular factorization of the matrix pencil b-za
[s,t,q,z] = reorder(s,t,q,z);   % reordering of generalized eigenvalues in ascending order

z21 = z(nk+1:end,1:nk);
z11 = z(1:nk,1:nk);

if rank(z11)<nk;
error('Invertibility condition violated')
end
```

```
z11i = z11\eye(nk);
s11 = s(1:nk,1:nk);
t11 = t(1:nk,1:nk);

if abs(t(nk,nk))>abs(s(nk,nk)) | abs(t(nk+1,nk+1))<abs(s(nk+1,nk+1));
warning('Wrong number of stable eigenvalues.');
end

dyn = s11\t11;

f = real(z21*z11i);      % The real function takes away very small imaginary parts of the
solution
p = real(z11*dyn*z11i);
```

13. reorder.m

```
function [s,t,q,z] = reordr(s,t,q,z)
n = size(s,1);
i = 1;
while i<=n-1;
if 1+abs(t(i,i)*s(i+1,i+1))>1+abs(s(i,i)*t(i+1,i+1));
[s,t,q,z] = qzswitch(i,s,t,q,z);
if ~(i==1);i = i-2;end
end
i=i+1;
end
```

Reference

Blanchard, Olivier & Charles Kahn, “The Solution of Linear Difference Models under Rational Expectations,” Econometrica 45, July 1985, 1305-1311.

Campbell J. “Inspecting the Mechanism: an Analytical Approach to the Stochastic Growth Model,” Journal of Monetary Economics, Vol 33, No3, 463-506.

Hansen G.D., “Indivisible Labor and the Business Cycle,” Journal of Monetary Economics, Vol.16, 309-327.

Kim, Jinnill & Sunghyun Henry Kim, “Spurious Welfare Reversals in International Business Cycle Models,” Journal of International Economics, forthcoming.

Schmitt-Grohe, S., and M. Uribe,“Solving Dynamic General Equilibrium Models Using a Second Order Approximation to the Policy Function,” Disscussion Paper, Rutgers University and University of Pennsylvania, 2002.

Sims, Christopher, “Solving Linear Rational Expectations Models,” Manuscript. Princeton: Princeton University, January 2000a (forthcoming Computational Economics).

Uhlig, Harald, “A Toolkit for Analyzing Nonlinear Dynamic Stochastic Models Easily,” 1995.