ACCURACY, SPEED AND ROBUSTNESS OF POLICY FUNCTION ITERATION*

Alexander W. Richter† Nathaniel A. Throckmorton‡ Todd B. Walker§

August 19, 2012

ABSTRACT

Policy function iteration methods for solving and analyzing dynamic, stochastic general equilibrium models are powerful from both a theoretical and computational perspective. Despite obvious theoretical appeal, significant startup costs and a reliance on a grid-based method have limited the use of policy function iteration as a solution algorithm. We reduce these costs by providing a user-friendly suite of MATLAB functions that introduce multi-core processing and Fortran via MATLAB's executable function. We demonstrate why policy function iteration is particularly useful in solving models with regime-dependent parameters, recursive preferences, and binding constraints. We examine a canonical real business cycle model and a new Keynesian model that features regime switching in policy parameters, Epstein-Zin preferences, and monetary policy that occasionally hits the zero-lower bound to highlight the attractiveness of our methodology. We compare our advocated approach to other familiar computational methods, highlighting the tradeoffs between accuracy and speed.

Keywords: Policy Function Iteration, Zero-Lower Bound, Epstein-Zin preferences

*We thank Troy Davig for many helpful discussions and Bulent Guler for helpful comments. Walker acknowledges support from the National Science Foundation under grant SES 096221.

†Department of Economics, Auburn University, arichter@auburn.edu

‡Department of Economics, Indiana University, nathrock@indiana.edu

§Department of Economics, Indiana University, walkertb@indiana.edu

1 INTRODUCTION

The Great Recession, the prospect of exponentially rising government debt, interest rates at the zero lower bound, and potential sudden changes to monetary and fiscal policy make clear that nonlinearities are a crucial element to contemporary macroeconomic analysis. Successfully modeling each of these scenarios requires large and persistent deviations from the non-stochastic equilibrium. Linear approximations around a deterministic steady state poorly captures these equilibrium properties.

We argue that policy function iteration is a flexible and accurate way to solve dynamic general equilibrium models with substantial nonlinearity. Policy function iteration methods for solving and analyzing dynamic, stochastic general equilibrium models are powerful from both a theoretical and computational perspective. Despite its theoretical appeal, significant startup costs and a reliance on a grid-based method have limited its use as solution algorithms. We reduce these costs by providing a user-friendly suite of MATLAB functions.

We demonstrate the usefulness of our approach by examining several topical examples. We begin with a simple real business cycle model and a standard New Keynesian model. These canonical models are useful starting points because their solutions and properties are well known. They also provide useful benchmarks for speed and accuracy. Section 3 provides step-by-step instructions for how to solve these models with policy function iteration in MATLAB using multi-core processing and integration of MATLAB and Fortran through executable files (MEX). Section 4 reports alternative methods for approximating the policy function, analyzing the tradeoffs between accuracy and speed. Using a 6-core processor (3.47GHz each), we find that our suite reduces computational time by a factor of 8 in the RBC model and by a factor of 24 in the NK model relative to non-parallelized code that does not utilize MEX. Additional stochastic components further increase the speeds gains associated with MEX and parallelization.

We then demonstrate the flexibility of policy function iteration. Section 6 introduces regime switching in monetary and fiscal policy parameters into the New Keynesian model with an emphasis on understanding the expectational effects generated by regime switching models. Sections 5 and 7 introduce Epstein-Zin preferences and a zero-lower bound on monetary policy into the New Keynesian model, respectively. We provide MATLAB code with extensive documentation for each example. Richter and Throckmorton (2012) provide additional supporting code.

The primary benefit of this algorithm over perturbation methods [Gaspar and Judd (1997); Judd and Guu (1992)] is its ability to easily account for sudden policy changes and other inherent nonlinearities (i.e. zero interest rate bound, default, irreversible investment, *etc.*). The flexibility and simplicity of the algorithm has led a recent segment of the literature to use this method to estimate monetary and fiscal policy regimes, quantify expectational effects of policy changes, and study key counterfactual policies [Bi (2011); Bi et al. (2011); Chung et al. (2007); Davig and Leeper (2006, 2008); Davig et al. (2010, 2011); Kumhof and Ranciere (2010)]. Policy function iteration and the suite of programs described in this paper can be easily adapted to handle models with: [i] endogenous regime change, [ii] temporarily nonstationary processes, [iii] binding collateral constraints, [iv] stochastic volatility, [v] news shocks, and [vi] heterogenous agents. Even though the methodology is grid-based, large models such as a two-country open economy with six continuous and one discrete state variable, and two stochastic processes can be solved in roughly one hour.

We remind readers that policy function iteration is a numerical byproduct of using monotone operators to prove existence and uniqueness of equilibria [Coleman (1991)]. This provides an

additional benefit to using policy function iteration as powerful approximation results and proofs of existence and uniqueness can be employed in conjunction with the numerical algorithm. We briefly review the theory of monotone operators in an online appendix.

2 PRACTICAL GUIDE

All of the routines required to implement the algorithm are written in MATLAB or compiled as MEX (Fortran 90) functions. This code and the code used to solve the models below are publicly available at `http://www.auburn.edu/~awr0007/`. The algorithm is broken down into a sequence of easily implementable steps that are executed with the following set of functions:

- `script.m` - Main script that establishes all user-specified inputs in the structure, `O`, and executes all functions. It contains a parallel for-loop that distributes the optimization routine at each point (node) in the discretized state space across all locally available processors, which considerably improves computation time. Optimization is performed with Chris Sims' `csolve.m` by finding updated policy functions that satisfy the equilibrium conditions of the model until a user-specified convergence criterion is met.
- `parameters.m` - Outputs a structure, `P`, containing the baseline calibration of the model.
- `steadystate.m` - Outputs a structure, `S`, with input `P`, containing the deterministic steady state and implied parameters of the model.
- `grids.m` - Outputs a structure, `G`, with inputs `O` and `P`, containing the discretized state space. The structure `O` contains the number of grid points and bounds of each state variable.
- `guess.m` - Outputs an initial conjecture for each policy function with inputs `O`, `P`, `S`, and `G`. The structure `O` specifies whether the initial conjectures for the policy functions are drawn from the log-linear solution to the model or the most recent iteration, which allows the user to resume the algorithm if it is interrupted prior to convergence.
- `variables.m` - Outputs a structure, `V`, containing an index of variables, forecast errors, and shocks and the corresponding variable descriptions.
- `linmodel.m` - Outputs the linear transition matrix, `T`, the impact matrix of the stochastic realizations, `M`, and a 2-element vector of flags, `eu`, indicating existence and uniqueness of the linear solution. The linear model is solved using Chris Sims' `gensys.m` algorithm and requires `P`, `S`, and `V` as inputs. `gensys.m` is user written MATLAB code that is designed to solve stochastic linear rational expectations models. The key to using this code is to map the model into the form

$$T_0 X_{t+1} = T_1 X_t + \Psi \varepsilon_{t+1} + \Pi \eta_{t+1},$$

where X is a vector of variables (exogenous and endogenous), ε is a vector of shocks, and η is a vector of forecast errors whose elements satisfy

$$\eta^x_{t+1} = x_{t+1} - E_t x_{t+1}$$

for some $x \in X$. Assuming the shocks are normally distributed and mean zero, `gensys` outputs the coefficients, T and M, of the following equation:

$$X_{t+1} = T X_t + M \varepsilon_{t+1}.$$

- `eqm.m` - Outputs a matrix, `R`, containing the residuals to a subsystem of expectational equations the are constrained by the remaining equations in the equilibrium system. This function requires as inputs the structures `P`, `S` and `G`, a vector of initial conjectures, the state at each node, and the current policy functions (required for interpolation/extrapolation).
- `allterp*.m` - Performs linear interpolation and extrapolation. MATLAB and Fortran routines (`Fallterp*.f90`) are provided. The naming convention uses a sequence of three integers representing the number of state variables, policy functions, and continuous stochastic variables. These functions can evaluate multiple points, but, in the interest of speed, are not generalized and may require straightforward modifications for alternative combinations of states, policies, and stochastic variables.[1]
- `pf.mat` - Data array containing the current policy functions and all structures.

2.1 NUMERICAL ALGORITHM This section outlines how to implement the algorithm. There are a number of model-specific initializations: model parameters, steady state values, and grid parameters (e.g. the number of points in each dimension and the distance between them) specified by the programmer. Through trial and error, we found that the initializations are best performed in the following order.

First, set the baseline calibration of the model in `parameters.m`. In addition, specify a convergence criterion for the policy functions, which affects the accuracy and speed of the solution.

Second, calculate (using a nonlinear solver, if necessary) the deterministic steady state and any implied parameters in `steadystate.m`. Steady state values are required for solving the linear model and are often helpful for setting up the discretized state space.

Third, specify the number of points and the bounds for each grid, which are contained in structure `O` and specified in `script.m`. The built-in MATLAB function `linspace` establishes a grid for each state variable in `grids.m`. The built-in MATLAB function `ndgrid` creates an array for each state variable where every position represents a unique permutation of the discretized state variables. The total number of elements in each array is the product of the number of points assigned to each state variable and represents the number of nodes in the discretized state space.

Fourth, establish an optimal set of variables to serve as numerical policies. The set of possible policy functions is not unique, but as a rule of thumb, establish policy functions over the minimum set of variables required to solve for all remaining time t variables given the state of the economy. We always reduce the number of policy variables to the number of equations with expectations operators.

Finally, obtain initial conjectures (guesses) for each policy function. Some models are extremely sensitive to the guess, but, in general, a linear solution provides a sufficient approximation.[2] We use Sims' (2002) `gensys.m` algorithm.[3] Enter the log- or level-linear system into `linmodel.m`. `guess.m` calls `linmodel.m` to solve the linear model and generate initial conjectures.

[1] Richter and Throckmorton (2012) provide the Fortran and MATLAB source code, as well as the compiled 32- and 64-bit MEX functions, for each of the examples discussed in the paper.

[2] This is not true for all models. For example, if the model contains discrete variables, such as state dependent parameters, then the conditional linear solution may poorly approximate the global solution. In this case, one can obtain a linear solution for each realization of the parameter(s) and a linear combination of those solutions typically provides a good guess for the state-dependent nonlinear model.

[3] Other methods include Uhlig's (1997) toolkit and Klein's (2000) algorithm.

After obtaining initial conjectures for each of the policy functions, set up the equilibrium system in `eqm.m`. Using the original policy function guesses or the solution from the previous iteration, solve for all time t variables using all of the equilibrium conditions, except those that contain expectations and require numerical integration. Next, calculate updated (time $t+1$) values for each of the policy variables using linear interpolation/extrapolation.[4] We provide two computational routines for this step—one written in MATLAB (`allterp*.m`) and one written in Fortran 90 (`Fallterp*.mex*`) but executed as a MATLAB executable (MEX) function. The Fortran code is much faster, but for those who are unfamiliar with MEX, the MATLAB function is sufficient for relatively small models—those with no more than four state variables and only one stochastic component. Further details about these routines are given in an online appendix.

After computing the updated values of each policy variable, solve for the remaining time $t+1$ variables needed to calculate time t expectations by applying numerical integration using the trapezoid rule or Gauss-Hermite quadrature. Additional details about these integration methods are provided in an online appendix. Finally, using Chris Sims' root finder, `csolve.m`, solve for the zeros of the equations with embedded expectations, subject to each of the remaining equilibrium conditions. The output of `csolve.m` on each node are policy values that satisfy the equilibrium system of equations to a specified tolerance level. This set of values characterizes the updated policy functions for the next iteration.

If the distance between the guess and the updated policy values is less than the convergence criterion on all nodes, then the policies have converged to their equilibrium values. Otherwise, use the updated policy functions as the new guess until convergence.

3 EXAMPLES

We first consider two conventional models—the real business cycle (RBC) model and the new Keynesian (NK) model with textbook treatments provided by McCandless (2008) and Walsh (2010).

3.1 RBC MODEL

The representative household chooses sequences, $\{c_{t+k}, k_{t+k}, n_{t+k}\}_{k=0}^{\infty}$, that maximize expected lifetime utility, $E_t \sum_{k=0}^{\infty} \beta^k \{c_{t+k}^{1-\sigma}/(1-\sigma) - \chi n_{t+k}^{1+\eta}/(1+\eta)\}$, where β is the subjective discount factor, $1/\sigma$ is the intertemporal elasticity of substitution, $1/\eta$ is the Frisch elasticity of labor supply, c is consumption, and n is labor hours. These choices are constrained by

$$c_t + k_t = w_t n_t + r_t^k k_{t-1} + (1-\delta)k_{t-1}$$
$$k_t = (1-\delta)k_{t-1} + i_t,$$

where w is the real wage, r^k is the real rental price of capital, k is the capital stock, and i is investment. The household's optimality conditions imply

$$1 = \beta E_t\{(c_t/c_{t+1})^{\sigma}(r_{t+1}^k + 1 - \delta)\} \quad \text{and} \quad w_t = \chi n_t^{\eta} c_t^{\sigma}.$$

Output is produced according to $y_t = z_t k_{t-1}^{\alpha} n_t^{1-\alpha}$, where $0 < \alpha < 1$. Productivity follows

$$z_t = (1-\rho)\bar{z} + \rho z_{t-1} + \varepsilon_t,$$

[4] Alternatively, Chebyshev polynomials or cubic splines could be used for interpolation/extrapolation. We typically opt for linear methods since they are faster and more stable, but we consider alternative methods in section 4.

where $\bar{z}$ is the average productivity level, and $\varepsilon \sim i.i.d.\ N(0, \sigma_\varepsilon^2)$. A perfectly competitive firm chooses $\{k_t, n_t\}$ to maximize $y_t - w_t n_t - r_t^k k_{t-1}$. The firm's optimality conditions imply

$$r_t^k = \alpha y_t / k_{t-1} \quad \text{and} \quad w_t = (1-\alpha) y_t / n_t.$$

The aggregate resource constraint is given by $c_t + i_t = y_t$. A competitive equilibrium is given by the household's and firm's optimality conditions, the production function, the law of motion for capital, the aggregate resource constraint, and the productivity process.[5]

3.1.1 SOLVING THE MODEL

The following are detailed instructions on how to setup and solve the nonlinear model described above.

- The model contains five structural parameters. Using a quarterly calibration, these parameters are set to $\beta = 0.99$ (4% real interest rate), $\delta = 0.025$, $\alpha = 0.33$, $\sigma = 1$, and $\eta = 1$. The productivity process is persistent with $\rho = 0.95$ and $\bar{z} = 1$. Productivity shocks are normally distributed with mean zero and standard deviation $\sigma_\varepsilon = 0.0025$. The convergence criterion is set to 1×10^{-10}.
- Given the steady state labor share, $\bar{n} = 0.33$, the deterministic steady state has a straightforward analytical solution.
- This model contains two continuous state variables, k_{t-1} and z_t, and one continuous stochastic variable, ε_{t+1}, that are discretized. The bounds of the state space and the number of points for each state variable are contained in the structure `O` and inputs of `grids.m`. The capital and productivity grids are evenly spaced with bounds that are 5% from their steady state values. These bounds minimize extrapolation and ensure that the model simulates on the discretized state space. With only two state variables, we can afford dense grids. We specify 41 points for each state variable (k_{t-1} and z_t), which implies 1,681 nodes. No specific number of grid points is required for any of the continuous state variables (minimum 3 points), but using more grid points on state variables that contribute to the curvature of the policy functions improves accuracy. However, there is delicate balance between increasing the number of grid points and keeping the problem numerically tractable.
- We enter the log-linear equilibrium (8 equations) into `linmodel.m` to produce initial conjectures to the nonlinear model.
- We choose n_t as a policy since it allows us to conveniently calculate the time t variables, but this choice is not unique. Given $\{k_{t-1}, z_t, n_t\}$, it is easy to calculate y_t from the production function, i_t from the aggregate resource constraint, k_t from the law of motion for capital, and c_t by combining the firm's and household's optimality conditions for labor. To obtain n_{t+1} interpolate/extrapolate for all realizations of ε_{t+1}. Given $\{k_t, z_{t+1}, n_{t+1}\}$, calculate c_{t+1} and r_{t+1}^k (from the firm's optimality condition), which enter expectations. We use Gauss Hermite quadrature to integrate across ε_{t+1}. `csolve.m` searches for labor policy values that satisfy the consumption Euler equation by setting the residual from `eqm.m` to zero.
- After entering the equilibrium system in `eqm.m`, execute `script.m` to run the algorithm. To take advantage of the speed gains associated with the parallel toolbox, execute the command `matlabpool`, which pools all locally available processors.

[5] We also provide code to solve the canonical RBC model with three conventional frictions—capital adjustment costs, variable utilization rates, and external habit persistence.

Table 1: RBC Solution Times–MEX, Parallelization Comparison* (in seconds)

	MATLAB		MEX	
Processors	Structures	No Structures	Structures	No Structures
1	129 (1.0)	100 (1.3)	90.5 (1.4)	62.9 (2.1)
2	65.6 (2.0)	52.4 (2.5)	47.1 (2.7)	34.2 (3.8)
6	27.5 (4.7)	23 (5.6)	20.7 (6.2)	16.6 (7.8)

* Based on a state space of 1,681 nodes (41 points on k_{t-1}, 41 points on z_t). We specify 10 realizations of ε_{t+1}. The routines were computed with an Intel Xeon X5690 6-core processor (3.47GHz) operating 64-bit Windows 7.

3.1.2 SOLUTION TIMES The major drawback of the algorithm is its reliance on a nonlinear solver that executes on each node, which is computationally expensive. The introduction of multi-core processing and Fortran via MEX reduces this computational burden (see table 1). Multi-core processing in MATLAB is easily facilitated with the Parallel Computing Toolbox (PCT), which creates a "worker pool" consisting of locally available processors. Without the PCT, MATLAB only uses one processor even though additional processors are available in most computers. In this case, computational resources are not maximized and each node in the discretized state space is evaluated sequentially. The PCT allows MATLAB to evaluate multiple nodes simultaneously, which produces near linear speed gains.

The interpolation/extrapolation step imposes the most significant slowdown. Our algorithm achieves further speed gains by using a Fortran 90 MEX function that is executed within MATLAB. For relatively small models (two or fewer state variables), full implementation using only MATLAB routines is relatively inexpensive. For larger models, such as the NK model, the benefits of MEX clearly outweigh the costs. The two pitfalls of programming in Fortran are the need for a MEX gateway function and the noninteractive nature of the editor. The MEX gateway function initializes the function's inputs and outputs and calls other subroutines that perform the actual task. There is a fixed cost in understanding how to initialize various functions with MATLAB's application programming interface, allocate memory, and create matrices and arrays that properly interact with MATLAB. An additional cost is the inconvenience of debugging. MATLAB's editor is proactive in addressing problem areas in scripts; Fortran is obviously not since it is a compiled language. Even if a Fortran function successfully compiles via MEX there still may be problems in the code that require further attention. Our Fortran functions are written using the free-format, which is relatively easy to learn since it is very similar to MATLAB syntax.[6] Once the Fortran function is compiled using MEX (which requires Intel Visual Fortran), a new MEX function is created, which is called in MATLAB in the exact same way as built-in functions.

MATLAB structures are a convenient way to group data—such as parameters, steady state values and grids—and reduce the number of inputs when calling functions. The drawback is that MATLAB requires more time to access values contained in structures Thus, convenience and simplicity are forfeited in favor of faster code by removing the values contained in the structures before the `parfor` loop.

[6]By default, MEX only interprets fixed-format (f77) Fortran code. Since our fortran code is written in the free-format (f90), MATLAB batch files need to be modified. Navigate to the mexopts folder in the MATLAB directory (e.g. . . . \MATLAB\R2010a\bin\win64\mexopts), and open the batch file corresponding to the installed version of Intel Visual Fortran (IVF) in a text editor. Delete the '/fixed' flag and save the batch file. Then use `mex -setup` to select IVF as the compiler in MATLAB.

Table 1 reports (in seconds) how the solution times differ across the three modifications introduced above—multi-core processing, interpolation via MEX, and MATLAB structures. Employing two processors instead of the one (default) reduces the solution time by approximately 50% regardless of the choice to use MEX or structures. Using MEX also cuts the solution time in half, which is independent of the number of processors or the use of structures. In both cases, adopting structures simplifies the code, but it imposes a fixed cost of roughly 30 seconds, which decreases with the number of processors.

The speed factor (in parentheses) represents the solution time relative to the slowest setup. Notice that it does not increase one-for-one with the number of processors, which is indicative of communication overhead. Additional processors still provide evident speed gains, but this benefit is limited if the code is not optimized for speed. For example, code that employs six processors without MEX while retaining structures yields nearly the same speed gain as code that employs just two processors, but uses MEX and removes the structures. Overall, using six processors with optimized code (MEX and no structures) reduces the solution time by a factor of nearly 8.

There are four main reasons why we typically choose not to implement the entire algorithm in Fortran. First, it maintains easily implementable code, since MATLAB is familiar to most economists. Second, it allows us to use readily available MATLAB functions, such as numerical solvers, instead of relying on the IMSL package, which imposes an additional cost. Third, parallelization is simple to implement with the PCT and does not require knowledge of the Fortran MPI or OpenMP. Finally, Fortran functions with several subroutines are often tedious to debug.

3.2 NEW KEYNESIAN MODEL

The representative household chooses sequences, $\{c_{t+i}, n_{t+i}, M_{t+i}, k_{t+i}\}_{i=0}^{\infty}$, that maximize expected lifetime utility,

$$E_t \sum_{i=0}^{\infty} \beta^i \left\{ \frac{c_{t+i}^{1-\sigma}}{1-\sigma} - \chi \frac{n_{t+i}^{1+\eta}}{1+\eta} + \nu \frac{(M_{t+i}/P_{t+i})^{1-\kappa}}{1-\kappa} \right\}, \qquad \chi, \nu > 0$$

where P_t is the aggregate price index, M_t is nominal money balances, and $1/\kappa$ is the interest (semi) elasticity of money demand. Following Dixit and Stiglitz (1977), $c_t \equiv [\int_0^1 c_t(i)^{(\theta-1)/\theta} di]^{\theta/(\theta-1)}$ is a consumption bundle composed of a continuum of differentiated goods, where $\theta > 1$ measures the price elasticity of demand. The agent's maximum attainable consumption bundle given a specific level of expenditures yields the demand function for good i, $c_t(i) = [p_t(i)/P_t]^{-\theta} c_t$, where $P_t = [\int_0^1 p_t(i)^{1-\theta} di]^{1/1-\theta}$. The household's choices are constrained by

$$c_t + m_t + i_t + b_t = (1-\tau_t)(w_t n_t + r_t^k k_{t-1}) + (m_{t-1} + r_{t-1} b_{t-1})/\pi_t + d_t$$
$$k_t = (1-\delta) k_{t-1} + i_t$$

where lower case letters denote real quantities ($x_t = X_t/P_t$). $\pi_t = P_t/P_{t-1}$ is the gross inflation rate, b_t is the stock of real government bonds, τ_t is a proportional tax rate levied against capital and labor earnings, and d_t is the share of real firm profits. Optimality implies

$$\chi n_t^{\eta} c_t^{\sigma} = (1-\tau_t) w_t$$
$$\nu m_t^{-\kappa} = (1 - 1/r_t) c_t^{-\sigma}$$
$$1 = \beta r_t E_t \{ (c_t/c_{t+1})^{\sigma} / \pi_{t+1} \}.$$
$$1 = \beta E_t \left\{ (c_t/c_{t+1})^{\sigma} \left((1-\tau_{t+1}) r_{t+1}^k + 1 - \delta \right) \right\}.$$

The production sector consists of monopolistically competitive intermediate goods producing firms who produce a continuum of differentiated inputs and a representative final goods producing firm. Each firm $i \in [0,1]$ in the intermediate goods sector produces a differentiated good, $y_t(i)$, with identical technologies given by $y_t(i) = k_{t-1}(i)^{\alpha} n_t(i)^{(1-\alpha)}$, where $k(i)$ and $n(i)$ are the levels of capital and employment used by firm i. Each intermediate firm chooses capital and labor to minimize its operating costs, $r_t^k k_{t-1}(i) + w_t n_t(i)$, subject to its production function.

The representative final goods producing firm purchases inputs from intermediate goods producing firms to produce a composite good, $y_t \equiv [\int_0^1 y_t(i)^{(\theta-1)/\theta} di]^{\theta/(\theta-1)}$. Maximizing profits for a given level of output yields firm i's demand function for intermediate inputs given by $y_t(i) = (p_t(i)/P_t)^{-\theta} y_t$. Following Rotemberg (1982), each firm faces a cost to adjusting its nominal price level, which emphasizes the potentially negative effect that price changes can have on customer-firm relationships. Using the functional form in Ireland (1997), real profits of firm i are

$$d_t(i) = \left[\left(\frac{p_t(i)}{P_t} \right)^{1-\theta} - \Psi_t \left(\frac{p_t(i)}{P_t} \right)^{-\theta} - \frac{\varphi}{2} \left(\frac{p_t(i)}{\bar{\pi} p_{t-1}(i)} - 1 \right)^2 \right] y_t,$$

where $\varphi \geq 0$ determines the magnitude of the adjustment cost, Ψ is real marginal costs, and $\bar{\pi}$ is the steady state gross inflation rate. Each intermediate goods producing firm chooses their price level, $p_t(i)$, to maximize the expected discounted present value of real profits $E_t \sum_{k=t}^{\infty} q_{t,k} d_k(i)$, where $q_{t,k} \equiv \prod_{k=t+1}^{T} q_{k-1,k}$ is the stochastic discount factor. In a symmetric equilibrium, all intermediate goods producing firms make the same decisions and the optimality condition reduces to

$$\varphi \left(\frac{\pi_t}{\bar{\pi}} - 1 \right) \frac{\pi_t}{\bar{\pi}} = (1-\theta) + \theta \Psi_t + \varphi E_t \left[q_{t,t+1} \left(\frac{\pi_{t+1}}{\bar{\pi}} - 1 \right) \frac{\pi_{t+1}}{\bar{\pi}} \frac{Y_{t+1}}{Y_t} \right].$$

In the absence of costly price adjustments (i.e. $\varphi = 0$), real marginal costs equal $(\theta - 1)/\theta$, which is equivalent to the inverse of the firm's markup factor.

The fiscal authority finances a constant level of discretionary spending, $\bar{g}$, through proportional taxes on capital and labor, seigniorage revenues, and by issuing nominal government debt. The government's flow budget constraint is given by

$$m_t + b_t + \tau_t (w_t n_t + r_t^k k_{t-1}) = \bar{g} + (m_{t-1} + r_{t-1} b_{t-1})/\pi_t.$$

Following Leeper (1991), the monetary and fiscal authorities set policy according to

$$r_t = \bar{r} (\pi_t / \pi^*)^{\phi} \quad \text{and} \quad \tau_t = \bar{\tau} (b_{t-1}/b^*)^{\gamma} \exp(\varepsilon_{\tau,t}),$$

where π^* and b^* are the target levels of debt and inflation, ϕ and γ are parameters controlling the policy responses to inflation and debt, and $\varepsilon_{\tau,t} \sim \ i.i.d. N(0, \sigma_\tau^2)$.

The aggregate resource constraint is given by $c_t + i_t + \bar{g} = [1 - \varphi(\pi_t/\bar{\pi} - 1)^2/2] y_t$. Equilibrium is characterized by the household's and firm's optimality conditions, the government's budget constraint, the monetary and fiscal policy rules, and the aggregate resource constraint.

3.2.1 SOLVING THE MODEL

The following instructions provide a complete description on how to apply the policy function iteration algorithm to this model.

Table 2: NK Model Solution Times* (in seconds)

Processors	No MEX	Partial MEX	All MEX
1	144.8 (1.0)	59.5 (2.4)	22.6 (6.4)
2	75.6 (1.9)	31.8 (4.6)	13.0 (11.1)
6	28.0 (5.2)	13.2 (11.0)	6.1 (23.7)

* Based on a state space of 2,401 nodes (7 points on each continuous state variable). We specify 10 realizations of ε_τ. The routines were computed with an Intel Xeon X5690 6-core processor (3.47GHz) operating 64-bit Windows 7. The value in parenthesis represents that speed factor increase from the slowest setup.

- The model contains seven structural parameters. Given an annual calibration, these parameters are set to $\beta = 0.9615$ (4% real interest rate), $\sigma = 1$, $\eta = 1$, $\kappa = 1$, $\theta = 7.66$ (15% price markup), $\delta = 0.1$, and $\varphi = 10$. The monetary policy parameter, $\phi = 1.5$, and the fiscal policy parameter, $\gamma = 0.15$, which guarantees a unique and bounded solution. Fiscal policy shocks are normally distributed with mean zero and standard deviation $\sigma_{\tau,t} = 0.001$.
- The preference parameters χ and ν are pinned down by the steady state labor share, $\bar{n} = 0.33$, and the velocity of money, $v = 3.8$. Given the steady state tax rate, $\bar{\tau} = 0.21$, and the share of government spending to output, $\bar{g}/\bar{y} = 0.17$, the remaining steady state values have a closed-form solution.
- Notice that the set of state variables, m_{t-1}, r_{t-1}, b_{t-1}, and k_{t-1}, can be reduced to a_t, b_{t-1}, and k_{t-1}, where $a_t \equiv m_{t-1} + r_{t-1}b_{t-1}$ is real government liabilities. It is also necessary to include the fiscal policy shock, $\varepsilon_{\tau,t}$, in the state to solve for the time t tax rate. Thus, the minimum state in the nonlinear model consists of $\{a_t, b_{t-1}, k_{t-1}, \varepsilon_{\tau,t}\}$.
- The NK model allows less flexibility for establishing policy functions. The presence of Rotemberg adjustment costs imply that inflation is a required policy to analytically solve for all time t variables. We specify labor as a policy function to pin down output. Additionally, we choose capital as a policy to minimize the number of computations. Given these policies and the state, all time t variables fall out naturally from the equilibrium conditions.
- Since this model contains four continuous state variables, it is too costly to specify grids with the same density as the RBC model. We specify seven points on each continuous state variable, which implies 2,401 nodes. The number of grid points is subjective, but the more dense the grid, the greater the accuracy of the solution. In models this size, we recommend first solving the model with relatively sparse grids and gradually increasing the density of the grids until there is no noticeable change in the policy functions.

3.2.2 SOLUTION TIMES Relative to the RBC model presented in section 3.1, the NK model contains two features that increase computational time. First, there are two additional state variables. With seven grid points on each continuous state variable, this increases the number of nodes in the discretized state space by about 40%. Second, there are two additional policy functions. The additional policy functions triple the amount of interpolation/extrapolation.

Table 2 reports speed test results across the number of cores and the amount of computations performed with MEX/Fortan. Once again, introducing multi-core processing produces near-linear speed gains regardless of the involvement of MEX. We report three different levels of MEX involvement—*No MEX* (none of the computations are performed using MEX), *Partial MEX* (only

the interpolation/extrapolation step is performed using MEX), and *All MEX* (the entire `eqm.m` function is performed using MEX). Performing the entire algorithm using MEX significantly reduces computational time. With six cores, *partial MEX* is roughly twice as fast as *No MEX*, but *All MEX* is another 2.5 times faster.

In total, multi-core processing and MEX reduces computational time from 144 to only 6 seconds—a speed factor of nearly 24. The reason for the significant improvement in speed gains over the RBC model is due to the `for`-loop in the interpolation/extrapolation step over two additional policies and the `for`-loop required by the non-linear solver, `csolve.m`.[7]

4 COMPARISON WITH ALTERNATIVE SOLUTION TECHNIQUES

Policy function iteration has nice theoretical properties that guarantee convergence to an equilibrium (see online appendix). This is because time iteration is a contraction mapping. The downside of time iteration is that it is costly from a computational perspective. Convergence can be slow relative to other solution procedures. This section examines alternative solution techniques.

We first solve the model using a log-linear approximation around its deterministic steady state. This method is very attractive because first-order Taylor approximations are straightforward to obtain, the solution to the equilibrium system is quick to compute, existence and uniqueness conditions are well established, and large models (in terms of the number of shocks, states, and policies) do not increase the computational burden. Moreover, these properties allow for estimation of model parameters. However, these benefits can come at a steep cost—inaccuracy outside of a close neighborhood around the deterministic steady state.

Linear interpolation locally approximates the policy functions at each node in the discretized state space. As an alternative to this local approximation method, we adopt fixed-point projection methods, which build global approximations of the policy functions, but still rely on a grid-based iterative technique [Judd (1992)]. Projection methods postulate that the policies can be written as a function of a user-specified basis, whose coefficients minimize the sum of squared residuals.

Given a particular set of least squares estimates (time $t-1$ policy function coefficients), fixed-point iteration uses both current and future values of the policy functions to back out the updated least squares estimates. In sharp contrast, time iteration only uses future values of the policy function and requires a nonlinear solver for the current policies. Moreover, fixed-point iteration does not permit parallelization or guarantee that the equilibrium system of equations is satisfied on each iteration to a specified tolerance level.

The accuracy of global solution methods depends on the choice of basis function. Given a chosen basis, each iteration implies new least squares estimates of the coefficients that globally approximate the policy function. We apply fixed-point projection methods with a monomial (FM) and Chebyshev polynomial (FC) basis. [Henceforth, we distinguish between methods using an acronym where the first letter is the iteration technique (time or fixed) and the second letter is the approximation method (linear interpolation, Chebyshev polynomial basis, monomial basis).] Orthogonal bases, such as Chebyshev polynomials, will yield more precise results, but greater precision comes at a greater computational cost, which is increasing in the order of the polynomial

[7] Solution times are not directly comparable across models because the speed of convergence may differ. For example, the NK model is larger in terms of the number of policy and the size of the discretized state space, but the running time of TL is nearly identical across models. This is because the RBC model takes four times as many iterations to converge.

used to approximate the policy functions.

Solving for the policy functions requires calculating the value of these functions off the nodes (i.e. interpolation between nodes and extrapolation outside the grid), which is inaccurate when there is curvature in the policy functions. As an alternative to TL, we interpolate using Chebyshev polynomials (TC), which better captures curvature. Like fixed-point projection methods, this alternative provides a better approximation of the policy function off the grid, but at a higher computational cost.

4.1 LEAST SQUARES PROJECTION

The following algorithm implements the fixed-point least squares projection method, given a specified basis. The basis is evaluated at a particular state and forms the dependent variables used to obtain the least squares estimates. Let X denote the basis evaluated using the original discretized state space and X_t the basis evaluated at the time t state.

We obtain an approximation of the policy functions, $\hat{\Lambda}_t$ (an M-by-p matrix where M is the number of nodes and p the number of policy functions), using the least-squares estimates, $\hat{\eta}_t$.[8] We use $\hat{\Lambda}_t$ to calculate the state variables at t, which forms X_t. $\hat{\Lambda}_{t+1}$ is a function of X_t and is used to calculate the $t+1$ variables necessary to evaluate the terms within the expectation operators. The expectations are evaluated using Gauss-Hermite quadrature.

Once the expectations are evaluated, we calculate Λ_t as implied by the equilibrium system of equations. Given Λ_t, new estimates of the coefficients are obtained using the least squares estimator, $\hat{\eta}_{t'}$. The algorithm iterates on these estimates. Since this method minimizes the sum of the squared residuals, the approximation error of the policy functions is minimized for a given basis. The least-squares estimates of the coefficients are updated using a convex combination of the old and new estimates, $\hat{\eta}_{t+1} = \lambda\hat{\eta}_{t'} + (1-\lambda)\hat{\eta}_t$ for $\lambda \in [0,1]$, which helps maintain stability, especially at the beginning of the algorithm. This iterative process continues until $|\hat{\eta}_{t'} - \hat{\eta}_t|$ reaches a pre-specified tolerance criterion.

4.1.1 MONOMIAL BASIS

Following Heer and Maussner (2005), we choose the set of monomials corresponding to an n^{th} order Taylor approximation of the policy function. The set of basis functions is given by

$$\mathcal{P}_n^s = \left\{ \left(x_1^{k_1} \cdots x_s^{k_s}\right) \middle| \sum_{i=1}^{s} k_i = j, k_i \geq 0, j = 0, 1, \ldots, p \right\},$$

where s is the number of state variables. The monomials are evaluated at each of the M nodes in the discretized state-space and stored column-wise in X. The approximated policy functions are given by $\hat{\Lambda}_t = X\hat{\eta}_t$. New estimates of the coefficients are obtained every iteration using the least squares estimator, $\hat{\eta}_{t'} = (X'X)^{-1}X'\Lambda_t$.

4.1.2 CHEBYSHEV POLYNOMIAL BASIS

As an alternative to the monomial basis, we approximate the policy functions using a Chebyshev polynomial basis. The first two Chebyshev polynomials are $T_0(x) = 1$ and $T_1(x) = x$. The remaining polynomials are given by the recursive formulation, $T_j(x) = 2xT_{j-1}(x) - T_{j-2}(x)$, for $j \geq 2$. We use the MATLAB function `ChebyshevCoeff.m`, written by David Terr, to obtain the coefficients for each polynomial. The

[8] We obtain the initial least-squares estimates from the log-linear solution.

MATLAB function `chebpoly.m` outputs these coefficients and the corresponding powers of x, which allows us to construct the polynomials with element-by-element matrix operations.

`chebpoly.m` also outputs the corresponding zeros of the Chebyshev polynomials. Discretizing the state-space so the nodes coincide with the zeros of the polynomials minimizes the error of the approximating function. The zeros of the Chebyshev polynomials are given by

$$\bar{x}_{k_i} \equiv \cos\left(\frac{2k_i - 1}{2m_i}\pi\right),$$

where $k_i = 1, \ldots, m_i$ and $i = 1, \ldots, s$. Choose m_i as the number of points in the i^{th} dimension of the state space. Since $\bar{x}_{k_i} \in [-1, 1]$, transform the grid to the interval $[a_i, b_i]$ by applying

$$\bar{z}_{k_i} \equiv \frac{\bar{x}_{k_i}(b_i - a_i)}{2} + a_i.$$

The next step is to estimate the least squares coefficients of the approximating function.

Define $\bar{\Lambda}_{k_1,\ldots,k_s}$ as the value of the approximating function, $\hat{\Lambda}(\bar{z}_{k_1}, \ldots, \bar{z}_{k_s})$. We seek $\eta_{j_1,\ldots,j_s}$ that minimizes

$$\sum_{k_1=1}^{m_1} \cdots \sum_{k_s=1}^{m_s} \left[\bar{\Lambda}_{k_1,\ldots,k_s} - \sum_{j_1=0}^{n_1} \cdots \sum_{j_s=0}^{n_s} \eta_{j_1,\ldots,j_s} T_{j_1}(\bar{x}_{k_1}) \cdots T_{j_s}(\bar{x}_{k_s})\right]^2$$

for $n_i \leq m_i$. This yields the least-squares estimator

$$\hat{\eta}_{j_1,\ldots,j_s} = \frac{1 + \mathbf{1}(j_1 > 0)}{m_1} \cdots \frac{1 + \mathbf{1}(j_s > 0)}{m_s} \sum_{k_1=1}^{m_1} \cdots \sum_{k_s=1}^{m_s} \bar{\Lambda}_{k_1,\ldots,k_s} T_{j_1}(\bar{x}_{k_1}) \cdots T_{j_s}(\bar{x}_{k_s})$$

and is the output of the MEX function `Fchebweights*`. The approximating function is evaluated in the MEX function `Fallcheb*` according to

$$\hat{\Lambda}(z_1, \ldots, z_s) = \sum_{j_1=0}^{n_1} \cdots \sum_{j_s=0}^{n_s} \hat{\eta}_{j_1,\ldots,j_s} T_{j_1}(x_1) \cdots T_{j_s}(x_s),$$

where $x_i = 2(z_i - a_i)/(b_i - a_i) - 1$.

4.2 EULER EQUATION ERRORS

Figure 1 compares errors for the consumption Euler equation across the log-linear, TL, TC, FM and FC solution methods. For ease of presentation, we show absolute errors in base 10 logarithms. This means that if the Euler equation error is -4, the household makes an error of one consumption good for every 10,000 units of consumption goods.

The log-linear method (circle markers) is the least accurate. FM (dashed line) relies on an arbitrarily-specified basis and is the least computationally intensive global approximation method. To capture nonlinearities in the policy functions, we write the approximating functions as $\Lambda = X\eta$, where $\Lambda = c_t$ is the sole policy function, $X = [1\ k\ z\ kz\ k^2\ z^2]$ is a collection of basis functions, and η is a 6×1 matrix of coefficients.[9] We find that this method is as much as two orders of magnitude more accurate than the log-linear method.

[9] We considered several alternative collections of basis functions with higher-order terms, but found that they had little effect on the magnitude of the errors.

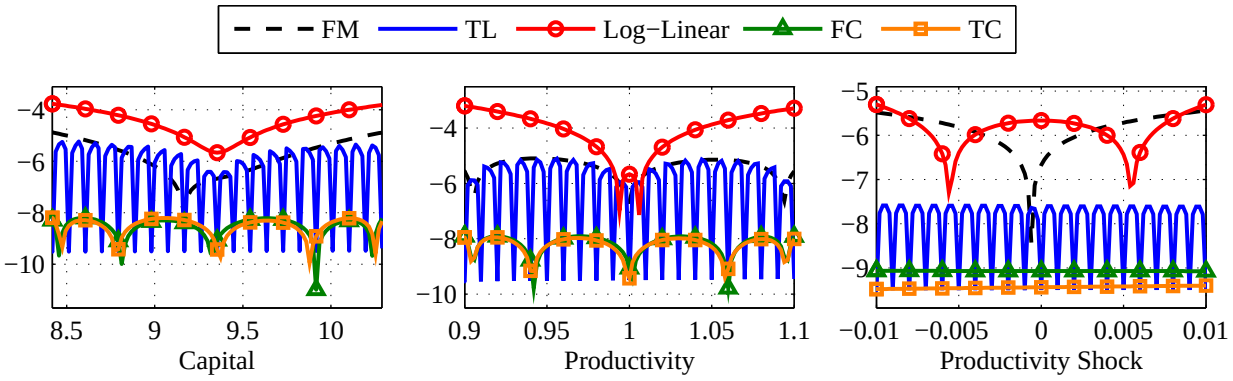


Figure 1: RBC model consumption Euler equation errors in base 10 logarithms. FM: Fixed point iteration with a monomial basis, TL: Time iteration with linear interpolation, FC: Fixed point iteration with a Chebyshev polynomial basis, TC: Time iteration with a Chebyshev polynomial basis. TC is based on a 4th order Chebyshev polynomial.

Like the log-linear method, TL (solid line) solves for a *local* approximation of the policy functions. Thus, the errors can meet any user-specified tolerance criterion on each node, but lose accuracy when policy function values are interpolated between nodes or extrapolated outside the state space. The key difference from the log-linear method is that TL outperforms FM off the grid for the productivity shock, improving the accuracy of the consumption Euler equation by as much as two orders of magnitude. FC and TC are even more accurate than TL. Regardless of the iteration technique, using Chebyshev polynomials to approximate the policy functions consistently satisfies the consumption Euler equation given any user-specified tolerance level—both on and off the grid. However, this result is sensitive to the order of the polynomial. The errors in figure 1 are based on a 4th-order Chebyshev polynomial. We have found that polynomials of third-order or smaller significantly reduce accuracy, but higher order polynomials only marginally improve accuracy.

The NK model contains five state variables—lagged real debt, real money balances, the nominal interest rate, capital, and the fiscal policy shock—and three policy functions—labor, inflation, and capital. We report errors for the consumption Euler equation, the firm pricing equation, and the bond Euler equation. For ease of presentation, we restrict our attention to errors as a function of capital and the fiscal policy shock.[10] Although this model is more complicated than the RBC model, the ordering of the Euler equation errors remains unchanged. Off the grid, TL consistently performs two orders of magnitude better than log-linear methods. TC and FC further increases accuracy, reducing Euler equation errors by an additional two orders of magnitude.

4.3 SPEED COMPARISONS Table 3 reports solution times for the RBC and NK models across the alternative solution methods. Linear methods are fast and easy to apply, and there are numerous toolboxes for obtaining solutions. However, these methods are less accurate and unable to handle intrinsic nonlinearities or capture potentially important expectational effects (for example, models with Markov-switching policy parameters or binding constraints.)

Figures 1 and 2 indicate that TC and FC offer a clear increase in accuracy (as much as two orders of magnitude), but table 3 shows that greater accuracy comes at the expense of a greater computational burden in terms of running time and implementation. Chebyshev interpolation is

[10] Alternative state variables do not impact the ordering of the Euler equation errors. We decided not to average across the linear state to obtain a better comparison between the linear and nonlinear solution techniques.

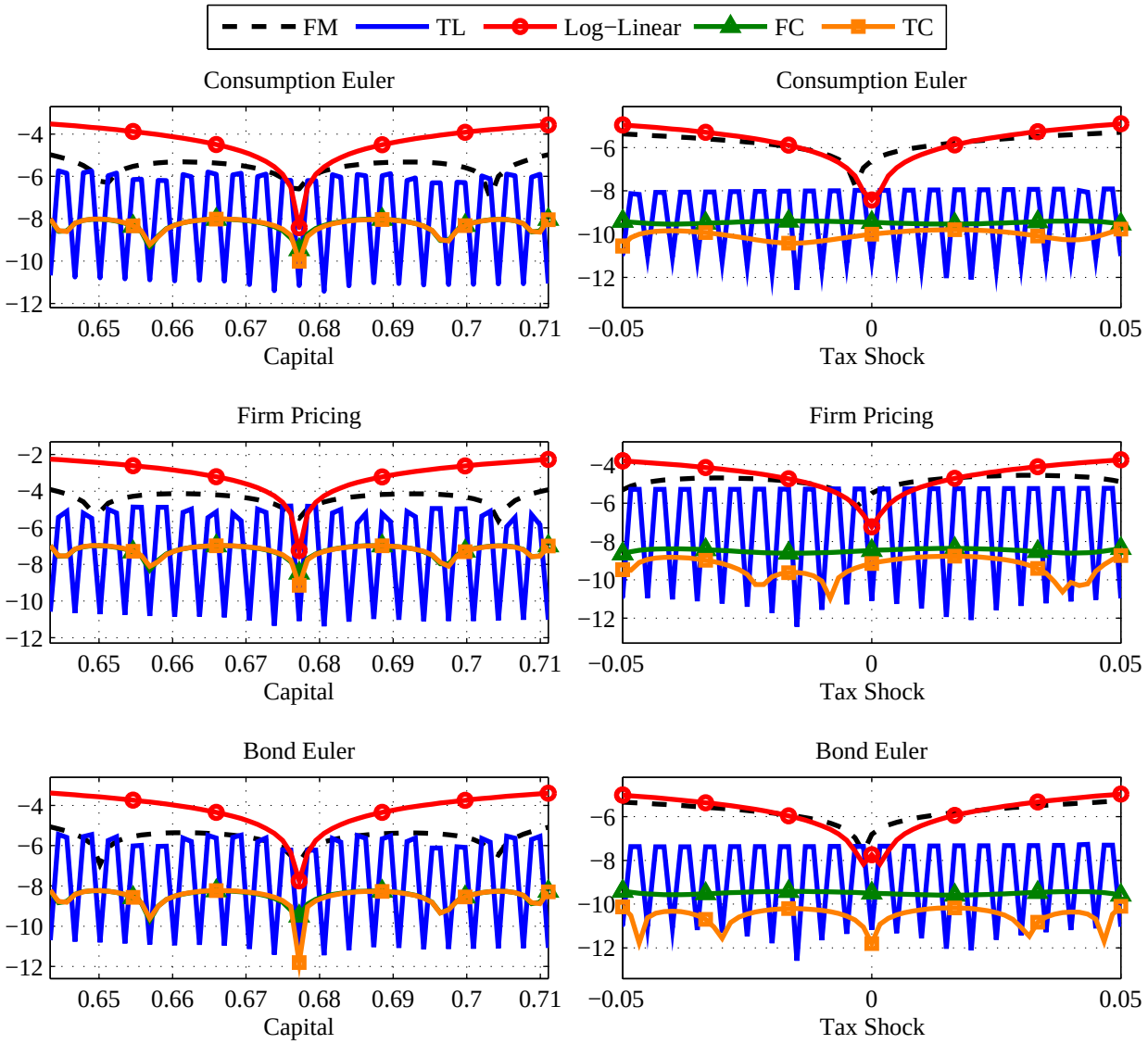


Figure 2: NK model Euler equation errors in base 10 logarithms. FM: Fixed point iteration with a monomial basis, TL: Time iteration with linear interpolation, FC: Fixed point iteration with a Chebyshev polynomial basis, TC: Time iteration with a Chebyshev polynomial basis. TC is based on a 4th order Chebyshev polynomial.

more intensive than linear interpolation, which MEX alleviates. The MEX implementation of FC is relatively quick and is four orders of magnitude more accurate than FM, but the speeds of both FC and TC relative to TL exponentially decrease for larger models and higher order Chebyshev polynomials. For the RBC and NK models, fixed-point is faster than time iteration. However, fixed-point solution times depend on the policy function update weight, λ. These simple models permit $\lambda = 1$, but complex models may require a lower λ to maintain stability of the algorithm. Time-iteration algorithms do not suffer from this instability since the nonlinear solver minimizes error each iteration. Our findings suggest that TL provides the best balance between speed and accuracy, offering a 41 percent (63 percent) speed decrease from TC in the RBC (NK) model with a modest reduction in accuracy. TL is also more robust than its alternatives, since it is stable in every macroeconomic model and calibration we have tested.

Table 3: Solution Times–Alternative Solution Methods Comparison* (in seconds)

	RBC		NK	
Method	MATLAB	MEX	MATLAB	MEX
FM	0.5	N/A	0.9	N/A
FC	27.2	6.5	302.0	31.2
TL	27.3	20.6	37.8	22.5
TC	45.9	23.9	508.2	60.8

* The RBC solution is based on a state space of 1,681 nodes (41 points on k_{t-1}, 41 points on z_t). We specify 10 realizations of ε_{t+1}. The NK solution is based on a state space of 2,401 nodes (7 points on a_{t-1}, b_{t-1}, k_{t-1}, and $\varepsilon_{\tau,t}$). We specify 10 realizations of $\varepsilon_{\tau,t+1}$. Speed tests for FC and TC are based on a 4th order Chebyshev polynomial and an update parameter (λ) equal to 1. The routines were computed with an Intel Xeon X5690 6-core processor (3.47GHz) operating 64-bit Windows 7. All 6 locally available processors were used in every speed test. FM: Fixed point iteration with a monomial basis, TL: Time iteration with linear interpolation, FC: Fixed point iteration with a Chebyshev polynomial basis, TC: Time iteration with a Chebyshev polynomial basis.

Table 4: RBC model Consumption Euler Equation Error Integrals (Maximums)*

	Capital	Productivity
FM	-8.4 (-4.9)	-8.7 (-5.1)
TL	-8.0 (-5.2)	-8.8 (-5.1)
Log-linear	-7.2 (-3.8)	-8.4 (-3.2)
FC	-10.3 (-8.2)	-11.3 (-7.9)
TC	-10.5 (-8.2)	-11.3 (-7.9)

* Values in base 10 logarithms. FM: Fixed point iteration with a monomial basis, TL: Time iteration with linear interpolation, FC: Fixed point iteration with a Chebyshev polynomial basis, TC: Time iteration with a Chebyshev polynomial basis. FC and TC are based on a 4th order Chebyshev polynomial.

4.4 MAX RESIDUAL AND EULER EQUATION ERRORS INTEGRAL

Following Aruoba et al. (2006), in tables 4 and 5 we provide the integral of the Euler equation errors and maximum residual for each solution method. These statistics provide complementary measures of accuracy and are defined over a square that is ± 10 percent on either side of steady state capital and productivity for the RBC model and ± 5 percent on either side of steady state capital and the tax shock for the NK model. We simulate each model for 100,000 periods from its stochastic steady state and create a distribution by counting the number of realizations in evenly spaced intervals and dividing by the length of the simulation.

The maximum Euler equation errors correspond to the largest errors plotted in figures 1 and 2. Once again, we report base 10 logarithms of the errors. There are two distinct groups. Listed from least to most accurate, the log-linear method, FM, and TL have the largest errors. FC and TC perform equally well and provide a clear improvement in accuracy.

The integrals take into consideration the frequency at which the errors occur and provide measures of the welfare loss associated with a particular solution method. Compared to the maximum residual, the relative accuracy of the various methods does not change. These integrals highlight a smaller difference in accuracy between TL and FM along the simulated path than suggested by the maximum error over the entire interval. However, FC and TC still show a clear comparative advantage in accuracy with at least a 4th order Chebyshev polynomial.

Table 5: NK Model Euler Equation Error Integrals (Maximums)*

	Capital			Tax Shock		
Method	Capital FOC	Firm Pricing	Bond FOC	Capital FOC	Firm Pricing	Bond FOC
FM	-9.5 (-5.0)	-8.4 (-3.9)	-9.8 (-5.1)	-9.4 (-5.3)	-8.3 (-4.5)	-9.6 (-5.3)
TL	-14.1 (-5.7)	-14.0 (-4.8)	-14.1 (-5.4)	-11.1 (-7.9)	-8.4 (-5.2)	-10.5 (-7.3)
Log-linear	-11.4 (-3.5)	-10.2 (-2.2)	-10.7 (-3.4)	-11.0 (-4.9)	-9.6 (-3.7)	-10.6 (-5.0)
FC	-12.4 (-8.0)	-11.4 (-7.0)	-12.4 (-8.2)	-12.2 (-9.4)	-11.2 (-8.4)	-12.3 (-9.4)
TC	-13.0 (-8.0)	-12.1 (-7.0)	-14.8 (-8.2)	-12.8 (-9.8)	-11.9 (-8.7)	-14.1 (-10.1)

* Values in base 10 logarithms. FM: Fixed point iteration with a monomial basis, TL: Time iteration with linear interpolation, FC: Fixed point iteration with a Chebyshev polynomial basis, TC: Time iteration with a Chebyshev polynomial basis. FC and TC are based on a 4th order Chebyshev polynomial.

5 NEW KEYNESIAN MODEL WITH RECURSIVE PREFERENCES

This section considers a cashless version of the NK model laid out in section section 3.2, but where households have preferences that distinguish between risk aversion and intertemporal substitution. This demonstrates the flexibility of policy function iteration and also shows how easily our algorithm can be adapted to fit a specific model. Following Giovannini and Weil (1989) and Epstein and Zin (1989, 1991), we adopt a recursive structure for intertemporal utility given by

$$U(u_t, E_t U_{t+1}^{1-\eta}) = \left\{(1-\beta)u_t^{(1-\eta)/\chi} + \beta(E_t U_{t+1}^{1-\eta})^{1/\chi}\right\}^{\chi/(1-\eta)}, \tag{1}$$

where η determines relative risk aversion, σ is the elasticity of intertemporal substitution, and $\chi = (1-\eta)/[1-\sigma^{-1}]$. Time-$t$ utility is given by

$$u_t \equiv u(c_t, n_t) = c_t^{\nu}(1-n_t)^{1-\nu}, \quad \nu \in (0,1). \tag{2}$$

The household's choices are constrained by

$$c_t + i_t + b_t = (1-\tau_t)(w_t n_t + r_t^k k_{t-1}) + r_{t-1} b_{t-1}/\pi_t + d_t \tag{3}$$

$$k_t = (1-\delta)k_{t-1} + i_t. \tag{4}$$

Given prices, the representative household chooses a sequence of quantities, $\{c_t, n_t, B_t, k_t\}$, to maximize (1) subject to (2)-(4). Optimality yields the following first order conditions

$$(1-\tau_t)w_t = \frac{1-\nu}{\nu}\frac{c_t}{1-n_t},$$
$$1 = r_t E_t\{Q_{t,t+1}/\pi_{t+1}\},$$
$$1 = E_t\{Q_{t,t+1}[(1-\tau_{t+1})r_{t+1}^k + (1-\delta)]\},$$

where the stochastic discount factor, Q, is given by

$$Q_{t,t+1} = \beta\left(\frac{u_{t+1}}{u_t}\right)^{\frac{1-\eta}{\chi}}\frac{c_t}{c_{t+1}}\left(\frac{V_{t+1}^{1-\eta}}{E_t V_{t+1}^{1-\eta}}\right)^{1-\frac{1}{\chi}}$$

and, at optimum, the value function can be written

$$V_t = \left\{(1-\beta)u_t^{(1-\eta)/\chi} + \beta[E_t V_{t+1}^{1-\eta}]^{1/\chi}\right\}^{\chi/(1-\eta)}.$$

When $\eta = 1/\sigma$, the value function, V_t, disappears from the first order conditions. Moreover, the constant of relative risk aversion, η, only alters the dynamics of higher order approximations, since, to a first-order, the term $(V_{t+1}^{1-\eta}/E_t V_{t+1}^{1-\eta})^{1-1/\chi}$ cancels out in expectation. The firm's problem and the policy specification are identical to section 3.2.

The model contains four state variables—lagged real debt, the nominal interest rate, capital, and the fiscal policy shock—and four policy functions—labor, inflation, capital, and the time-t value of the optimal value function. Figure 3 reports errors for the consumption Euler equation, the bond Euler equation, the firm pricing equation, and the household's optimal value function. Once again, we restrict our attention to errors as a function of capital and the fiscal policy shock and report them in base 10 logarithms.

Outside of a close neighborhood of steady state, the log-linear method continues to perform at least two orders of magnitude worse than its nearest nonlinear competitor. However, the accuracy of the FM solution technique increases relative to the canonical NK model, performing equally as well as the TL solution method. TC and FC remain the most accurate, improving accuracy by an additional two orders of magnitude over FM and TL.

6 NEW KEYNESIAN MODEL WITH REGIME SWITCHING

This section adds monetary and fiscal policy switching to the canonical NK model, but where the fiscal authority only has access to lump-sum taxes to isolate the expectational effects of switching regimes. The monetary and tax rules are

$$r_t = \bar{r}(\pi_t/\pi^*)^{\phi(s_t)}\exp(\varepsilon_{r,t})$$
$$\tau_t = \bar{\tau}(b_{t-1}/b^*)^{\gamma(s_t)}\exp(\varepsilon_{\tau,t}),$$

where $s_t = \{1, 2\}$ and the regime-dependent reaction coefficients are given by

$$\phi(s_t) = \begin{cases}\phi & \text{for } s_t = 1,\\ 0 & \text{for } s_t = 2,\end{cases} \qquad \gamma(s_t) = \begin{cases}\gamma & \text{for } s_t = 1,\\ 0 & \text{for } s_t = 2.\end{cases}$$

The policy mix evolves according to a first-order two-state Markov chain given by

$$\begin{bmatrix}\Pr[s_t = 1|s_{t-1} = 1] & \Pr[s_t = 2|s_{t-1} = 1]\\ \Pr[s_t = 2|s_{t-1} = 1] & \Pr[s_t = 2|s_{t-1} = 2]\end{bmatrix} = \begin{bmatrix}p_{11} & p_{12}\\ p_{21} & p_{22}\end{bmatrix}.$$

Following Leeper (1991), we label state 1 as active monetary and passive fiscal (AM/PF) policy and state 2 as passive monetary and active fiscal (PM/AF) policy.

6.1 SOLUTION TECHNIQUE AND EULER EQUATION ERRORS

The two policy regimes imply very different linear solutions. Thus, using only one of the linear solutions does not provide a good initial conjecture for both policy states. Instead, we use a linear combination of the two linear solutions with weights equal to the transition probabilities to form a guess for the full nonlinear

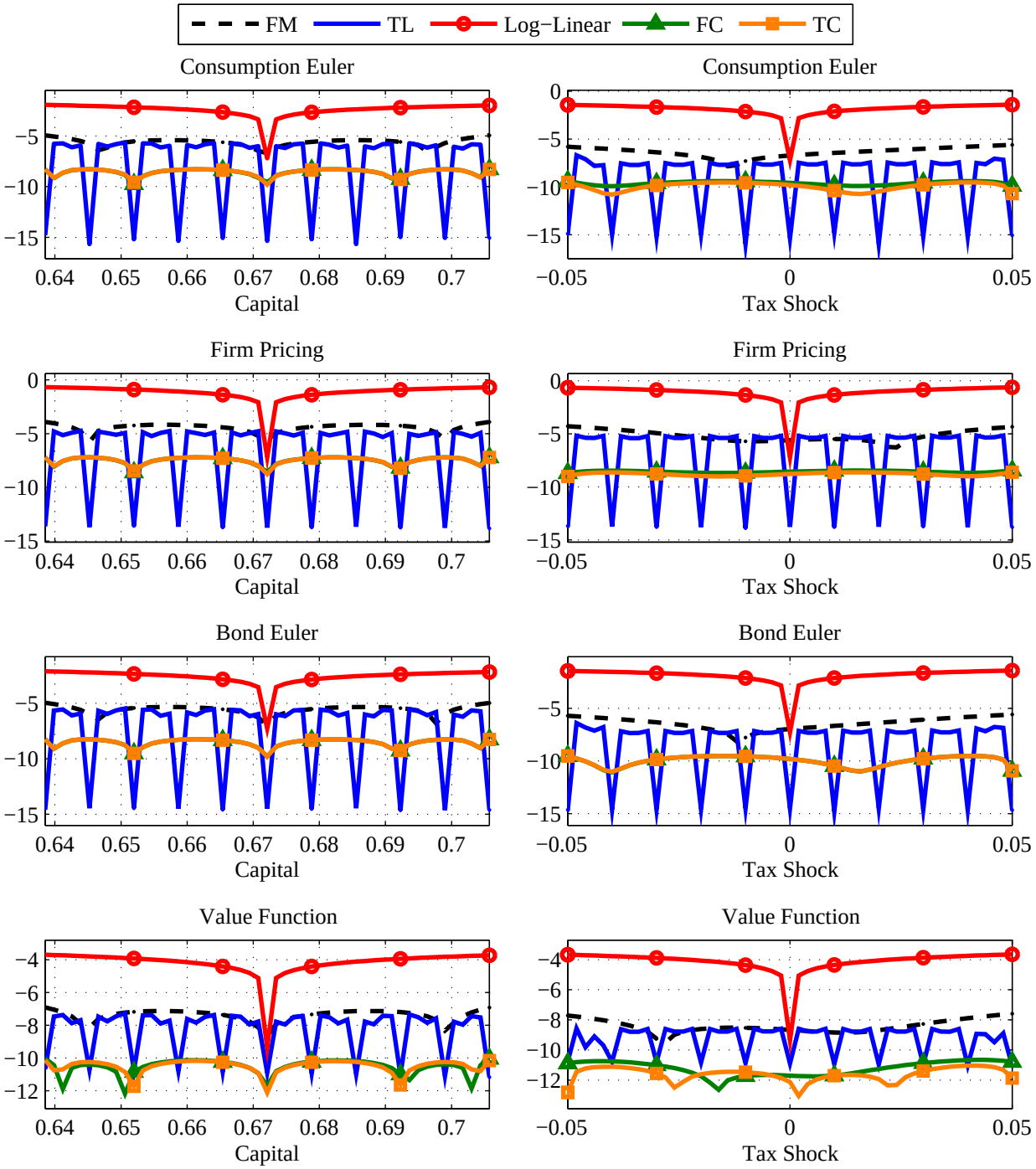
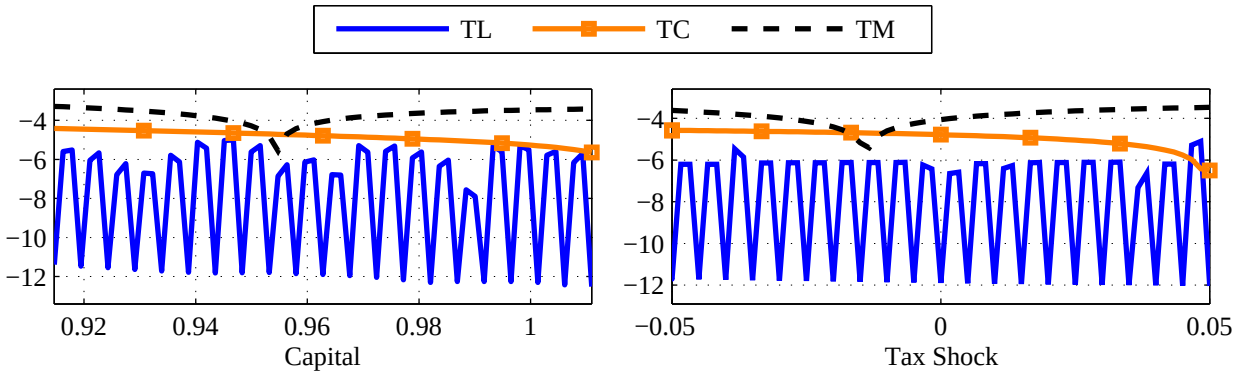


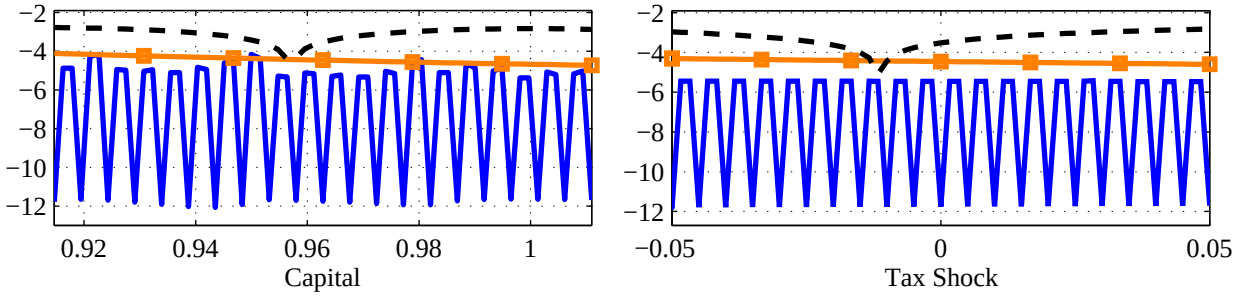
Figure 3: NK model with recursive preferences Euler equation errors in base 10 logarithms. FM: Fixed point iteration with a monomial basis, TL: Time iteration with linear interpolation, FC: Fixed point iteration with a Chebyshev polynomial basis, TC: Time iteration with a Chebyshev polynomial basis.

model. We have found that this is the best approach because it approximates the expectational effect of switching between regimes.

We interpolate/extrapolate for every permutation of the discretized continuous stochastic variables, $\varepsilon_{\tau,t}$ and $\varepsilon_{r,t}$. We facilitate this with a nested loop in `allterp*` that outputs a matrix of all



(a) Active Monetary/Passive Fiscal ($s_t = 1$)



(b) Passive Monetary/Active Fiscal ($s_t = 2$)

Figure 4: NK model with regime switching in the policy parameters consumption Euler equation errors in base 10 logarithms. TL: Time iteration with linear interpolation, TC: Time iteration with a Chebyshev polynomial basis, TM: Time iteration with a monomial basis. TC is based on a 4th order Chebyshev polynomial and $p_{11} = p_{22} = 0.8$.

possible realizations of the policy variables at time $t + 1$, where the rows correspond to the values of $\varepsilon_{\tau,t}$ and the columns correspond to the values of $\varepsilon_{r,t}$.

To evaluate expectations, we perform numerical integration using a two-step process that first applies the Trapezoid rule to each continuous stochastic variable and subsequently applies Markov-Chain integration to the discrete stochastic variable. To integrate across the continuous stochastic variables, first replicate the vector of $\varepsilon_{\tau,t}$ weights across the m realizations of $\varepsilon_{r,t}$ to create an $n \times m$ weighting matrix. Then weight each expectation and apply the Trapezoid rule across the $\varepsilon_{\tau,t}$-dimension (the rows) to collapse each expectation to a vector of realizations. Finally, weight each of these outcomes using the $\varepsilon_{r,t}$ weights and once again apply the Trapezoid rule. This step produces expectations conditional on the realizations of the discrete stochastic variable, s_t. To integrate across each of these outcomes, simply weight each conditional expectation by its likelihood and sum across all realizations.

Figure 4 compares the accuracy of the TM, TL, and TC solutions to the New Keynesian model with switching in the policy parameters.[11] We specify $p_{11} = p_{22} = 0.8$. We plot the residuals for the AM/PF ($s_t = 1$) and PM/AF ($s_t = 2$) regimes, which are roughly the same. With a 4th-order polynomial, TC is less accurate than TL. This is in sharp contrast with our findings in the non-switching NK model, where Chebyshev interpolation is consistently more accurate than linear

[11] We only provide the consumption Euler equation errors, since the firm pricing and bond Euler equation errors imply the same qualitative results. The other errrors are available from the authors upon request.

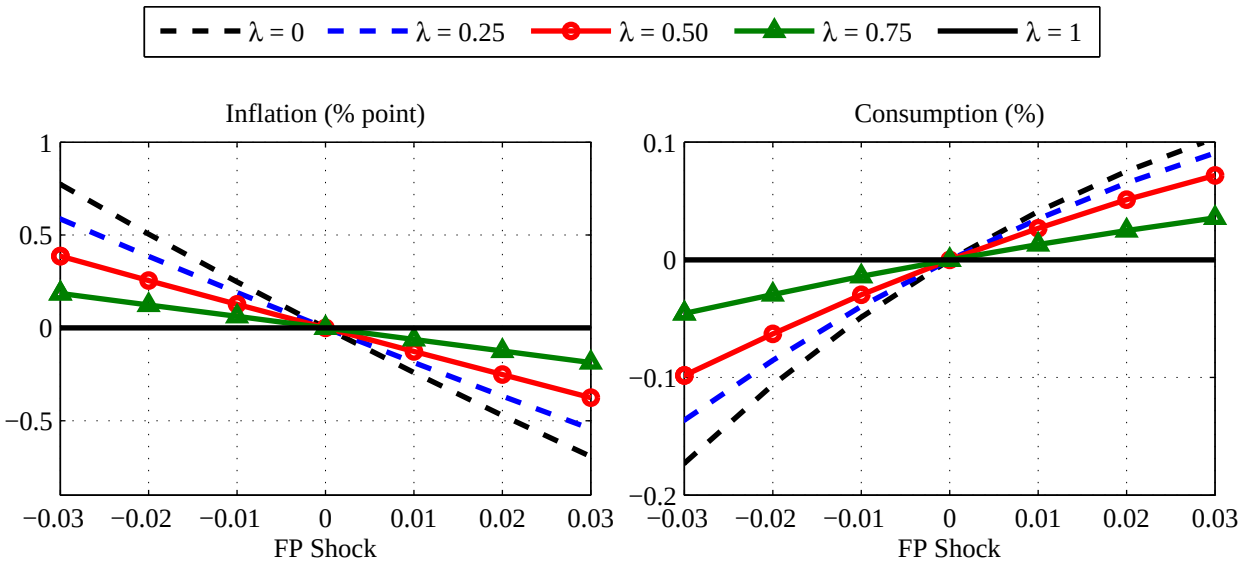


Figure 5: Policy functions for the NK model with regime switching in the policy parameters, based on a time iteration/linear interpolation technique (TL). λ measures the average duration of time spent in the active monetary/passive tax regime (AM/PF, $s_t = 1$) in the ergodic distribution.

interpolation. Chebyshev interpolation is a global method and cannot capture the discontinuity as well as linear interpolation, which is a local method.

6.2 EXPECTATIONAL EFFECTS

The effect of monetary and fiscal policy shocks is dependent on the probabilities in the transition matrix. The average duration of time spent in the active monetary/passive tax regime (AM/PF, $s_t = 1$) in the ergodic distribution is given by

$$\xi = \frac{1 - p_{22}}{2 - p_{11} - p_{22}}.$$

Figure 5 shows how the policy functions in the AM/PF regime change as ξ declines, which demonstrates the nonlinearities that the TL solution method captures. In the passive monetary/active fiscal regime (PM/AF, $s_t = 2$), a tax shock elicits no response from the fiscal authority and the price level adjusts to stabilize debt. When $\xi = 1$, there is no chance of moving to the PM/AF regime. Thus, Ricardian equivalence holds and shocks to taxes have no effect on the policies. As the expected duration of time spent in the PM/AF regime rises, negative tax shocks increase inflation, which, with costly price adjustments, lowers output and consumption.

TL is particularly attractive for solving models with discrete random variables. As figure 5 illustrates, this method captures the curvature in the policy functions, even with linear interpolation. Spectral methods increase accuracy, but rely on smoothness in the policy functions and are far less stable and less accurate when applied to models with regime-switching. We were unable to achieve convergence using TC when $\xi < 0.5$ or when the polynomial order exceeds 4.

7 ZERO LOWER BOUND ON THE NOMINAL INTEREST RATE

This section explains how the solution procedure changes when the economy faces a zero lower bound (ZLB) on the nominal interest rate (henceforth, the ZLB algorithm). We augment the simple

Taylor rule in the New Keyesian model to include an automatic stabilizer component, and given the inequality constraint on the interest rate, the monetary policy rule becomes

$$r_t = \max(1, \bar{r}(\pi_t/\pi^*)^{\phi_\pi}(y_t/\bar{y})^{\phi_y}).$$

To simplify the analysis, we assume the economy is at its cashless limit and that the government balances its budget each period by levying lump-sum taxes (government debt is in zero net supply). The economy is subject to preference and productivity shocks. The preference shock propagates through the discount factor and productivity augments production. They evolve according to

$$\beta_{t+1} = \beta(\beta_t/\beta)^{\rho_\beta}\exp(\varepsilon_\beta),$$
$$a_{t+1} = a(a_t/a)^{\rho_a}\exp(\varepsilon_a),$$

where $\varepsilon_x \sim N(0, \sigma_x^2)$, $x \in \{\beta, a\}$.

7.1 BASIC SOLUTION TECHNIQUE The ZLB algorithm combines both time and fixed-point iteration techniques. The policy functions for the NK model—labor, inflation, and capital—are updated in two steps. The first step follows the time iteration procedure described in section 2 without imposing the ZLB. This step obtains an optimal inflation rate corresponding to the current guess for the policy functions. The second step imposes the ZLB and uses the Euler equations to calculate implied values for labor and capital. This two-step approach is appealing for the following reasons. First, the time iteration step maintains stability of the algorithm since it solves for an inflation rate that minimizes the Euler equations given the current guess for the policy functions. Second, fixed-point iteration can handle an occasionally binding constraint, whereas time iteration cannot since it depends on a Newton-Raphson solver. Finally, combining the two techniques is a straightforward extension to solving the model without the ZLB.

7.2 SOLVING THE MODEL The following are detailed instructions on how to setup and solve the canonical NK model with a ZLB that occasionally binds.

- The deep parameters are identical to section 3.2.1 with the exception of the preference parameter, $\beta = 0.99$ (1% real interest rate). The productivity and discount rate processes are semi-persistent with autoregressive coefficients, $\rho_a = \rho_\beta = 0.6$. The shocks are normally distributed with mean zero and standard deviations $\sigma_a = 0.01$ and $\sigma_\beta = 0.005$. The monetary authority response coefficients, ϕ_π and ϕ_y, are set to 1.5 and 0.6. We normalize steady state output to 1 and solve for the implied steady state productivity level.

- Both the linear and nonlinear models contain a unique set of state variables—capital, productivity, and the preference parameter. We discretize each state variable with 21 points, which implies a total of 9,261 nodes. Once again, the number of grid points is subjective, but given the curvature of the policy functions, denser grids are necessary. There is also flexibility when choosing the policy functions. We specify policies over inflation (required), labor, and capital.

- We obtain initial conjectures of the policy functions from the solution to the nonlinear model where the ZLB does *not* bind. Although this solution implies negative nominal interest rates on some of the nodes and does not make sense economically, it serves as a good guess for the ZLB algorithm.

- To solve the model with a binding ZLB, we use a two-step process to update the policy functions. First, use time iteration to solve for the policy function values that minimize the Euler equation errors to the equilibrium system without imposing the ZLB. Unlike the fixed-point algorithm, the time iteration algorithm does not assume the current policy functions are true at time t, and instead minimizes the Euler equation errors using a nonlinear solver, which improves stability. Without imposing the ZLB, this procedure updates the policy functions on every node even if the interest rate implied by the Taylor rule is less than one.

  Second, determine where the interest rate binds, given the optimal policy functions calculated in the first step. Calculate the interest rate from the ZLB Taylor rule on each node. Output is a function of the discretized productivity and capital state variables and the updated labor policy function from the time iteration step. The interest rate is a function of the updated inflation policy function and output. If the ZLB does *not* bind, use the optimal policy functions from the first step as a guess on those nodes until convergence. If the ZLB binds, set $r_t = 1$ and update the policy functions using fixed-point iteration to back out the implied values of labor and capital from the Euler equations. More specifically, we first use the optimal inflation policy function values from the time iteration step and labor and capital policy function values from the previous iteration to update the state variables and interpolate. Then we evaluate expectations and use the Euler equations to update the labor and capital policy functions where the ZLB binds. To facilitate this process, we created a new function called `pf_up.m`, but all other functions remain the same.

- Each iteration the policy functions are updated as a linear combination of old and new values using the update weight, λ, to improve stability of the algorithm. We specify $\lambda = 0.2$. Continue this two step process until the policy functions meet a specified convergence criterion on every node—those where the ZLB binds and does not bind.

7.3 ZLB RESULTS To study the consequences of interest rates at their zero lower bound, it is important that the ZLB binds on some the nodes in the state space. Figure 6 shows the productivity-discount rate cross-section of the state space where the ZLB binds. We fix capital at 4.5% above steady state to concentrate on a region of the state space where the ZLB occasionally binds and there is curvature in the policy functions. We report these regions for the TL and TC solutions methods. Both higher productivity and more patient households increase output and drive the nominal interest rate to its ZLB. Given our discretized state space and model calibration, the ZLB binds on roughly 7% of the nodes for TL and roughly 10.6% of the nodes for TC. Wider bounds on any of the state variables will increase this percentage.

Figure 7 reports errors for the consumption Euler equation, firm pricing equation, and bond Euler equation, as a function of the capital, productivity, and the discount rate states in base 10 logarithms. To report errors for a region of the state space where the ZLB occasionally binds, each state not represented on the horizontal axis is fixed at 4% above steady state. Similar to section 6.1, TL outperforms TC with a 3rd order polynomial, but both methods are less accurate when the ZLB binds (asterisks mark nodes where the ZLB binds). A higher order Chebyshev polynomial may achieve a more accurate solution, but there is a tradeoff between accuracy and stability. We were unable to achieve stability with a Chebyshev polynomial order above 3 on all state variables.

Figure 8 shows the consumption and labor policy functions obtained from the TL solution method. Once again, we fix capital at 4.5% above steady state and the nodes where the state space

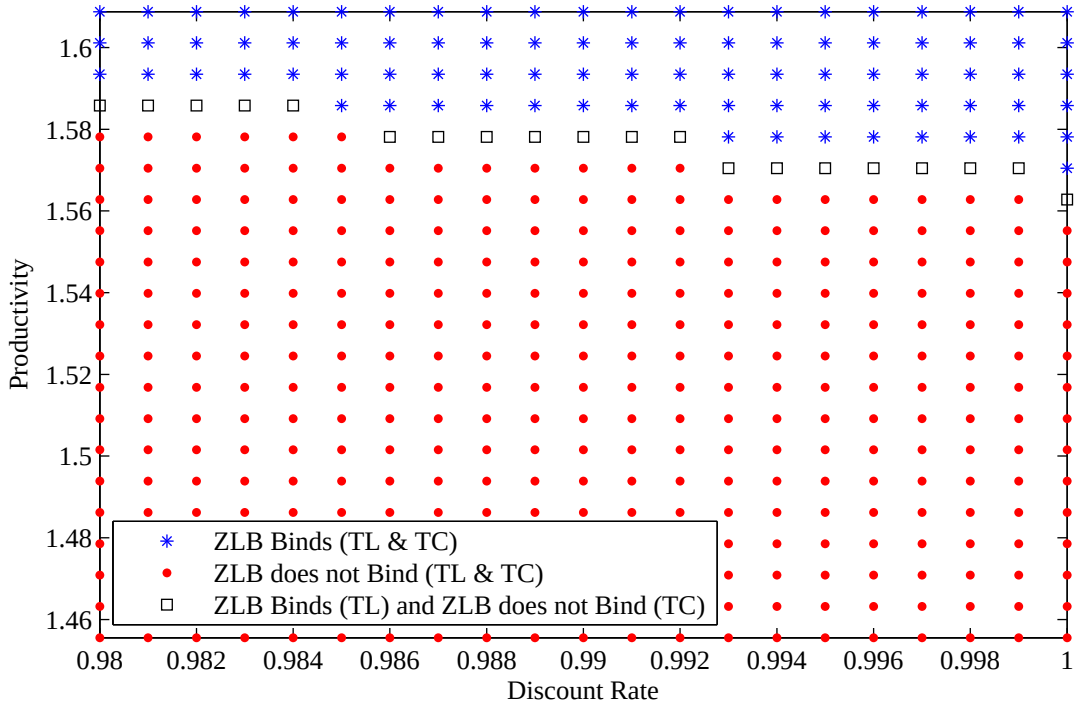


Figure 6: Productivity-discount rate cross-section of the state space where capital is 4.5% above steady state.

binds are marked by asterisks. The inequality constraint induces significant curvature in the policy functions that does not exist in other NK models, even when they contain real frictions. At the ZLB, households save more than they otherwise would have since the interest rate is higher than the one implied by the optimal inflation rate, which induces them to work more and consume less.

8 CONCLUSION

Policy function iteration has long been known as a reliable way to solve dynamic models nonlinearly. It is particularly useful for studying the economic consequences of a wide variety of potential policy outcomes. Despite its considerable benefits, this algorithm suffers from several drawbacks. The most prohibitive feature of this algorithm is its reliance on a grid-based technique, which exponentially increases the size of the problem with the number of state variables. Perturbation solution methods are far less computationally expensive and are more appropriate for models that do not contain recurring regime change or equilibrium paths that deviate far from the deterministic steady state. Another concern is whether this solution method consistently satisfies transversality conditions, since it only iterates on the policy functions and has no formal mechanism for imposing these restrictions. As a safeguard, however, it is easy to simulate a model for thousands of periods and check that its average asset levels (e.g. capital, bonds, *etc.*) are convergent. Moreover, simulated paths in models that explicitly violate the transversality condition will typically diverge even if the algorithm converges. Although these exercises do not provide proof, they do provide reasonable confidence that transversaility conditions are met. Lastly, in complicated models that include several policy regimes, it is often difficult to obtain initial conjectures that lead to convergence. In these cases, the best approach is to use the linear solutions to obtain nonlinear guesses

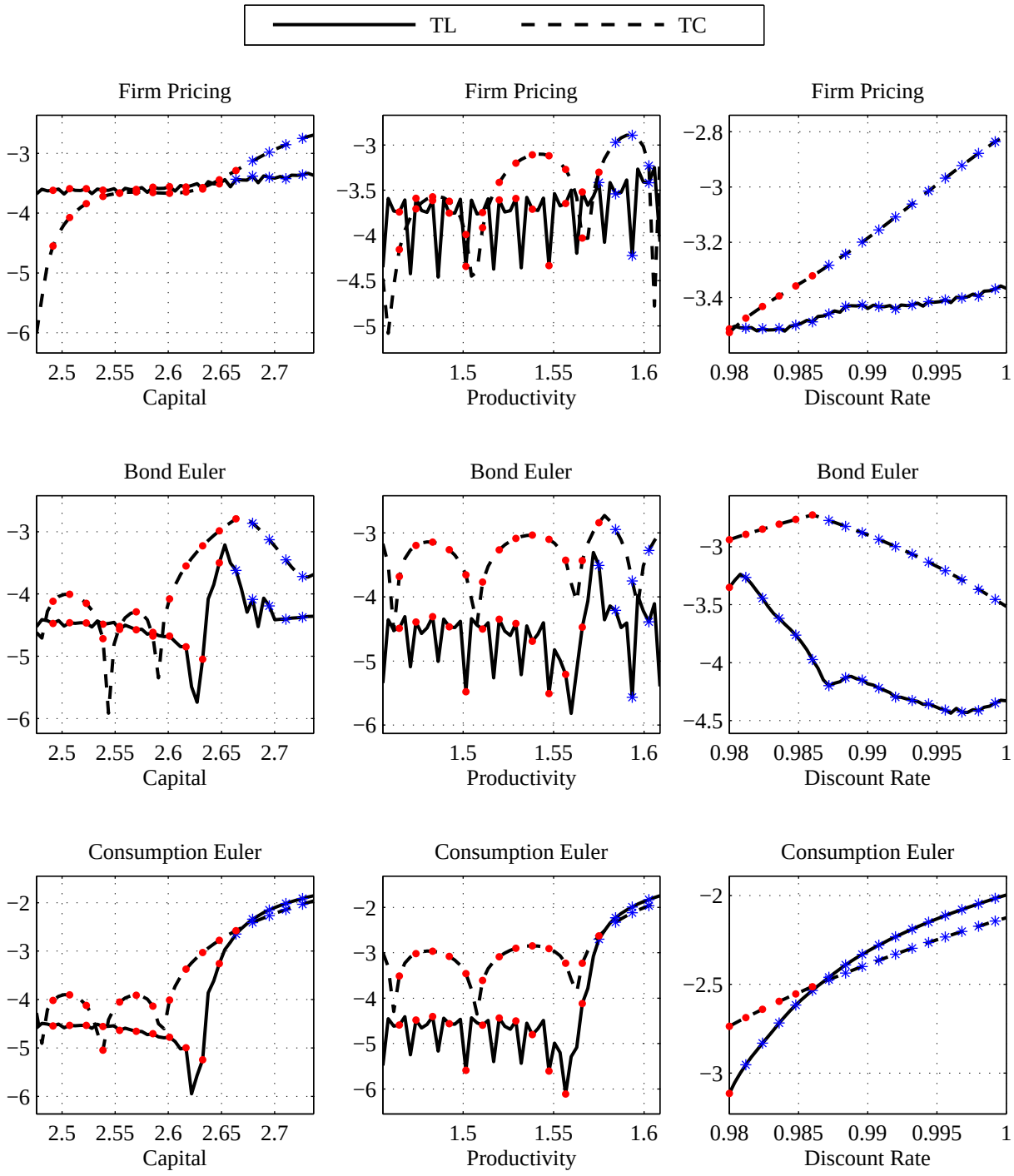


Figure 7: NK ZLB model Euler equation errors in base 10 logarithms. Each state not represented on the horizontal axis is fixed at 4% above steady state. TL: Time iteration with linear interpolation, TC: Time iteration with a Chebyshev polynomial basis. TC is based on a 3rd order Chebyshev polynomial. Asterisks mark nodes where the ZLB binds and points marks nodes where the ZLB does not bind.

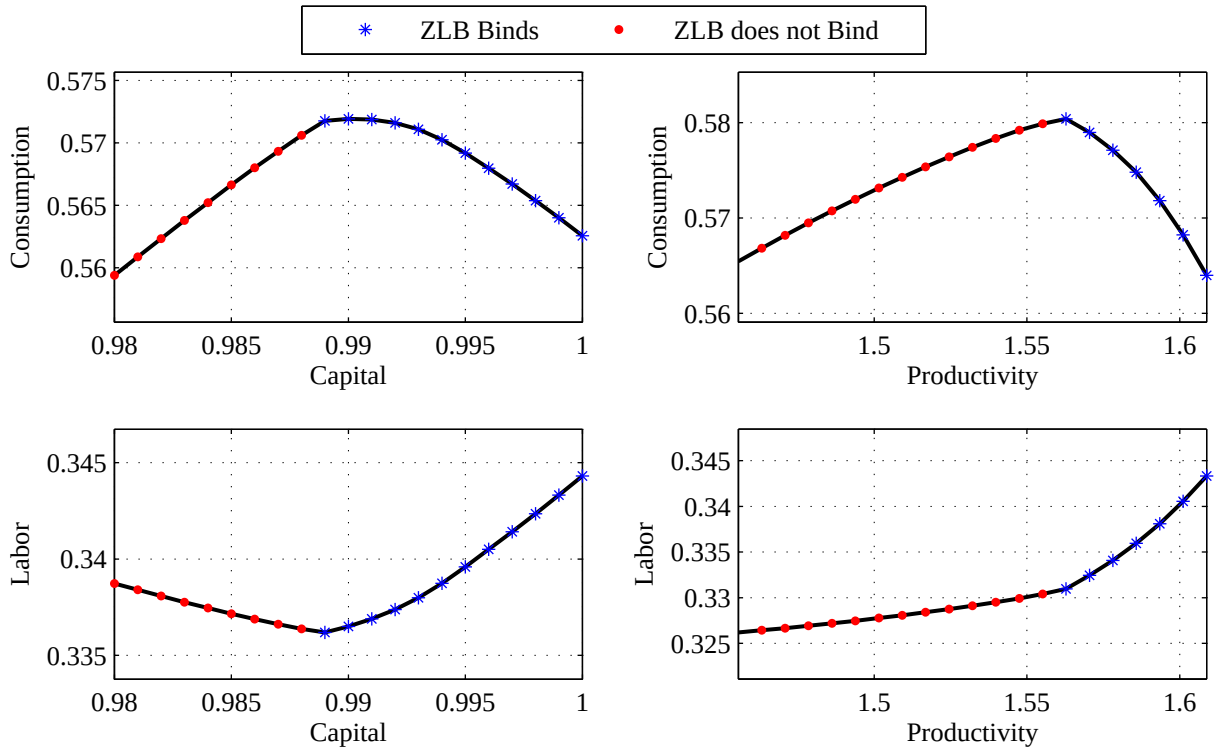


Figure 8: Policy functions with an occasionally binding zero-lower bound obtained from the TL solution method. Capital is fixed at 4.5% above steady state.

for each policy regime and weight them according to their transition probabilities.

Any numerical algorithm imposes direct costs onto the programmer. We reduce the costs associated with policy function iteration by providing a user-friendly suite of MATLAB functions that introduce multi-core processing and Fortran via MATLAB's executable function. We apply the algorithm to conventional RBC and NK models and carefully document how to chose policy functions, discretize the state space, interpolate/extrapolate future values, and perform numerical integration. The use of multi-core processing alongside optimized code that takes advantage of Fortran's comparative advantage at evaluating loops decreases solutions times by a factor of 8 in the RBC model and a factor of 24 in the NK model. Moreover, comparing time iteration with linear interpolation to alternative solution techniques demonstrates it is accurate and able to capture important nonlinearities.

REFERENCES

ARUOBA, S. B., J. FERNANDEZ-VILLAVERDE, AND J. F. RUBIO-RAMIREZ (2006): "Comparing solution methods for dynamic equilibrium economies," *Journal of Economic Dynamics and Control*, 30, 2477–2508.

BAXTER, M. (1991): "Approximating suboptimal dynamic equilibria : An Euler equation approach," *Journal of Monetary Economics*, 28, 173–200.

BAXTER, M., M. CRUCINI, AND K. ROUWENHORST (1990): "Solving the stochastic growth model by a discrete-state-space, Euler-equation approach," *Journal of Business & Economic Statistics*, 8, 19–21.

BI, H. (2011): "Sovereign Default Risk Premia, Fiscal Limits and Fiscal Policy," Bank of Canada Working Paper 2011-10.

BI, H., E. M. LEEPER, AND C. LEITH (2011): "Uncertain Fiscal Consolidations," Manuscript, Indiana University.

CHUNG, H., T. DAVIG, AND E. M. LEEPER (2007): "Monetary and Fiscal Policy Switching," *Journal of Money, Credit and Banking*, 39, 809–842.

COLEMAN, II, W. J. (1991): "Equilibrium in a Production Economy with an Income Tax," *Econometrica*, 59, 1091–1104.

DATTA, M., L. MIRMAN, O. MORAND, AND K. REFFETT (2005): "Markovian Equilibrium in Infinite Horizon Economies with Incomplete Markets and Public Policy," *Journal of Mathematical Economics*, 41, 505–544.

DATTA, M., L. MIRMAN, AND K. REFFETT (2002): "Existence and Uniqueness of Equilibrium in Distorted Dynamic Economies with Capital and Labor," *Journal of Economic Theory*, 103, 377–410.

DAVIG, T. (2004): "Regime-switching Debt and Taxation," *Journal of Monetary Economics*, 51, 837–859.

DAVIG, T. AND E. M. LEEPER (2006): "Fluctuating Macro Policies and the Fiscal Theory," in *NBER Macroeconomics Annual 2006, Volume 21*, ed. by D. Acemoglu, K. Rogoff, and M. Woodford, MIT Press, Cambridge, 247–298.

——— (2008): "Endogenous Monetary Policy Regime Change," in *NBER International Seminar on Macroeconomics 2006*, ed. by L. Reichlin and K. D. West, MIT Press, Cambridge, 345–391.

DAVIG, T., E. M. LEEPER, AND T. B. WALKER (2010): "'Unfunded Liabilities' and Uncertain Fiscal Financing," *Journal of Monetary Economics, Carnegie-Rochester Conference Series on Public Policy*, 57, 600–619.

——— (2011): "Inflation and the Fiscal Limit," *European Economic Review, Special Issue on Monetary and Fiscal Interactions in Times of Economic Stress*, 55, 31–47.

DIXIT, A. K. AND J. E. STIGLITZ (1977): "Monopolistic Competition and Optimum Product Diversity," *American Economic Review*, 67, 297–308.

EPSTEIN, L. G. AND S. E. ZIN (1989): "Substitution, Risk Aversion, and the Temporal Behavior of Consumption and Asset Returns: A Theoretical Framework," *Econometrica*, 57, 937–69.

——— (1991): "Substitution, Risk Aversion, and the Temporal Behavior of Consumption and Asset Returns: An Empirical Analysis," *Journal of Political Economy*, 99, 263–86.

GASPAR, J. AND K. JUDD (1997): "Solving Large-Scale Rational-Expectations Models," *Macroeconomic Dynamics*, 1, 45–75.

GIOVANNINI, A. AND P. WEIL (1989): "Risk Aversion and Intertemporal Substitution in the Capital Asset Pricing Model," NBER Working Paper 2824.

GREENWOOD, J. AND G. W. HUFFMAN (1995): "On the Existence of Nonoptimal Equilibria in Dynamic Stochastic Economies," *Journal of Economic Theory*, 65, 611–623.

HEER, B. AND A. MAUSSNER (2005): *Dynamic General Equilibrium Modelling: Computational Methods and Applications*, Springer.

IRELAND, P. N. (1997): "A Small, Structural, Quarterly Model for Monetary Policy Evaluation," *Carnegie-Rochester Conference Series on Public Policy*, 47, 83–108.

JUDD, K. L. (1992): "Projection methods for solving aggregate growth models," *Journal of Economic Theory*, 58, 410 – 452.

JUDD, K. L. AND S.-M. GUU (1992): "Perturbation Solution Methods for Economic Growth Models," in *Economic and Financial Modeling with Mathematica*, ed. by H. R. Varian, Springer, New York, 80–103.

KLEIN, P. (2000): "Using the Generalized Schur Form to Solve a Multivariate Linear Rational Expectations Model," *Journal of Economic Dynamics and Control*, 24, 1405–1423.

KUMHOF, M. AND R. RANCIERE (2010): "Inequality, Leverage and Crises," IMF Working Paper 10-268.

LEEPER, E. M. (1991): "Equilibria Under 'Active' and 'Passive' Monetary and Fiscal Policies," *Journal of Monetary Economics*, 27, 129–147.

MCCANDLESS, G. (2008): *The ABCs of RBCs: An Introduction to Dynamic Macroeconomic Models*, Cambridge: Harvard University Press.

MIRMAN, L., O. MORAND, AND K. REFFETT (2008): "A Qualitative Approach to Markovian Equilibrium in Infinite Horizon Economies with Capital," *Journal of Economic Theory*, 139, 75–98.

RICHTER, A. AND N. THROCKMORTON (2012): "MATLAB/MEX/Fortran Toolbox," Available online. http://auburn.edu/~awr0007/AWR_files/MatlabMEXtoolbox.zip.

ROTEMBERG, J. J. (1982): "Sticky Prices in the United States," *Journal of Political Economy*, 90, 1187–1211.

SIMS, C. A. (2002): "Solving Linear Rational Expectations Models," *Computational Economics*, 20, 1–20.

UHLIG, H. (1997): "A Toolkit for Analyzing Nonlinear Dynamic Stochastic Models Easily," Manuscript, Center for Economic Research.

WALSH, C. E. (2010): *Monetary Theory and Policy, Third Edition*, vol. 3, Cambridge, Massachusetts: The MIT Press.

A ONLINE APPENDIX (NOT FOR PUBLICATION)

A.1 BRIEF REVIEW OF THE THEORY

This section briefly reviews the theory behind monotone operators as applied to DSGE models, using the results of Greenwood and Huffman (1995) (GH, hereafter). We do not convey any new theoretical results but simply demonstrate how the monotone map can be used to prove existence and uniqueness of an equilibrium. We follow GH closely because they proved existence of equilibrium in a very general setup. Moreover, as advocated by Datta et al. (2002), Datta et al. (2005), Mirman et al. (2008), the theoretical properties of the monotone map can be extended to more complex setups. Proofs of existence using monotone operators are constructive in the sense that the numerical algorithm is a byproduct of the proof, which only adds to the appeal of the policy function iteration algorithm.

A.1.1 ECONOMIC ENVIRONMENT

The economic environment is standard. The model consists of a continuum of measure one identical agents with preferences

$$E_0\left[\sum_{t=0}^{\infty}\beta^t U(c_t)\right],$$

where the momentary utility function is assumed to be strictly increasing, strictly concave and twice differentiable, with $U'(0)=\infty$. The production technology is given by

$$y_t = F(k_t, K_t, \eta_t),$$

where output, y_t, is produced with the individual agent's capital stock, k_t, the aggregate capital stock, K_t, and is subject to a random technology shock, η_t. The technology shock is assumed to be drawn from a Markov distribution function, $G(\eta_{t+1}|\eta_t)$, with bounded support. The production function is assumed to satisfy the Inada conditions, $\lim_{K\to 0} F_1(K,K,\eta)=\infty$, be strictly increasing and strictly concave in its first argument, and twice differentiable in its first two arguments. Moreover, GH also impose the following somewhat nonstandard assumptions:

1. $\exists \bar{K} \ni F(\bar{K},\bar{K},\eta)\leq \bar{K}$
2. $\forall K\in(0,\bar{K}],\ F_1(K,K,\eta)+F_2(K,K,\eta)\geq 0 \quad$ and $\quad F_{11}(K,K,\eta)+F_{21}(K,K,\eta)<0$

Assumption 1 places an upper bound on the level of output. Assumption 2 requires that the sum of the marginal products of the individual and aggregate capital stock be positive (along the equilibrium path $k=K$). As noted by GH, these assumptions are innocuous and hold for a wide range of economies.

The agent's dynamic programming problem is given by

$$V(k,K,\eta)=\max_{k'}\left\{U(F(k,K,\eta)-k')+\beta\int V(k',K',\eta')dG(\eta'|\eta)\right\} \tag{5}$$

where aggregate capital, K, has the following law of motion $K'=Q(K,\eta)$. Let the optimal policy function associated with (5) be given by $k'=q(k,K,\eta)$. By standard arguments, one can derive the corresponding Euler equation

$$U'(F(k,K,\eta)-k')=\beta\int U'(F(k',K',\eta')-k'')F_1(k',K',\eta')dG(\eta'|\eta)$$

A stationary equilibrium is a pair of functions, $k' = q(k, K, \eta)$ and $K' = Q(k, \eta)$, that satisfy optimality (i.e., solves (5)) and consistency, $q(K, K, \eta) = Q(K, \eta)$. We are now able to state the main proposition of GH.

Proposition 1 (GH, pg. 615)**.** *There exists a nontrivial stationary equilibrium for the economy described above.*

The method of proof in GH follows that of Coleman (1991) and is our primary interest because it uses Euler equation iteration and properties of monotone operators. For these reasons we repeat the proof here. Let the sequence of aggregate laws of motion, $\{H^j(K, \eta)\}_{j=0}^{\infty}$, evolve according to $H^0(K, \eta) \equiv 0$, and let $H^{j+1}(K, \eta)$ for $j \geq 0$ be defined as the solution for x in the Euler equation

$$U'(F(K, K, \eta) - x) = \beta \int U'(F(x, x, \eta') - H^j(x, \eta'))F_1(x, x, \eta')dG(\eta'|\eta). \tag{6}$$

(6) defines a sequential operator mapping H^j into H^{j+1}. GH show that the left-hand side of (6) is strictly increasing in x, while the right-hand side is strictly decreasing in x.[12] This monotonic mapping along with assumptions 1 and 2 imply the existence of a solution to (6).

The intuition behind the result is straightforward: the sequence $\{H^j(K, \eta)\}_{j=0}^{\infty}$ produces a monotonically increasing sequence for the aggregate capital stock, which is bounded above by $\bar{K}$. GH prove that the pointwise limit of this sequence of functions is the aggregate policy function $\lim_{j\to\infty} H^j(K, \eta) = Q(K, \eta)$ and that the aggregate law of motion is nondegenerate (i.e., a degenerate law of motion is one that satisfies $Q(K, \eta) = F(K, K, \eta)$ for all K and η).

This mapping serves as the basis for numerical algorithms discussed in this paper, among many others [Coleman (1991); Baxter (1991); Baxter et al. (1990); Davig (2004); Davig et al. (2010); Bi (2011)]. While the purpose of this paper is to provide resources to reduce the cost of implementing the computational algorithm down, the theoretical aspect of the monotone map is very appealing.

A.2 LINEAR INTERPOLATION/EXTRAPOLATION

To get an idea of how linear interpolation and extrapolation works, consider the following example with two state variables, x_1 and x_2. The nearest perimeter around the point, (x_1', x_2'), is formed by the four points $(x_{1,i}, x_{2,i})$, $(x_{1,i}, x_{2,i+1})$, $(x_{1,i+1}, x_{2,i})$, and $(x_{1,i+1}, x_{2,i+1})$, where i signifies the position on the grid. We want the policy function value, $f(x_1', x_2')$, but we only have policy function values for the four nearest points on the grid (off the grid, we extrapolate using the nearest four points that form a square on the edge of the state space). First, holding x_2 fixed, interpolate/exptrapolate in the x_1 direction to obtain

$$\begin{aligned} f(x_1', x_{2,i}) &= f(x_{1,i}, x_{2,i}) + (x_1' - x_{1,i})\frac{f(x_{1,i+1}, x_{2,i}) - f(x_{1,i}, x_{2,i})}{x_{1,i+1} - x_{1,i}} \\ &= \frac{x_{1,i+1} - x_1'}{x_{1,i+1} - x_{1,i}} f(x_{1,i}, x_{2,i}) + \frac{x_1' - x_{1,i}}{x_{1,i+1} - x_{1,i}} f(x_{1,i+1}, x_{2,i}) \end{aligned} \tag{7}$$

$$f(x_1', x_{2,i+1}) = \underbrace{\frac{x_{1,i+1} - x_1'}{x_{1,i+1} - x_{1,i}}}_{\omega_{1,i}} f(x_{1,i}, x_{2,i+1}) + \underbrace{\frac{x_1' - x_{1,i}}{x_{1,i+1} - x_{1,i}}}_{\omega_{1,i+1}} f(x_{1,i+1}, x_{2,i+1}). \tag{8}$$

[12] In order to prove the right-hand side is strictly decreasing, the additional assumption, $0 \leq \partial H^j(K, \eta)/\partial K \leq [F_1(K, K, \eta) + F_2(K, K, \eta)]$, needs to be imposed.

Then interpolate/exptrapolate in the x_2 direction to obtain

$$\begin{aligned} f(x_1', x_2') &= f(x_1', x_{2,i}) + (x_2' - x_{2,i})\frac{f(x_1', x_{2,i+1}) - f(x_1', x_{2,i})}{x_{2,i+1} - x_{2,i}} \\ &= \underbrace{\frac{x_{2,i+1} - x_2'}{x_{2,i+1} - x_{2,i}}}_{\omega_{2,i}} f(x_1', x_{2,i}) + \underbrace{\frac{x_2' - x_{2,i}}{x_{2,i+1} - x_{2,i}}}_{\omega_{2,i+1}} f(x_1', x_{2,i+1}). \end{aligned} \tag{9}$$

Combining (7), (8) and (9) yields

$$\begin{aligned} f(x_1', x_2') &= \omega_{2,i}\left(\omega_{1,i} f(x_{1,i}, x_{2,i}) + \omega_{1,i+1} f(x_{1,i+1}, x_{2,i})\right) \\ &\quad + \omega_{2,i+1}\left(\omega_{1,i} f(x_{1,i}, x_{2,i+1}) + \omega_{1,i+1} f(x_{1,i+1}, x_{2,i+1})\right) \\ &= \omega_{1,i}\omega_{2,i} f(x_{1,i}, x_{2,i}) + \omega_{1,i+1}\omega_{2,i} f(x_{1,i+1}, x_{2,i}) \\ &\quad + \omega_{1,i}\omega_{2,i+1} f(x_{1,i}, x_{2,i+1}) + \omega_{1,i+1}\omega_{2,i+1} f(x_{1,i+1}, x_{2,i+1}) \\ &= \sum_{j_1=0}^{1}\sum_{j_2=0}^{1} \omega_{1,i+j_1}\omega_{2,i+j_2} f\left(x_{1,i+j_1}, x_{2,i+j_n}\right), \end{aligned}$$

which can be easily extended to any number of state variables. We assume that the points for any one dimension in the state space are uniformly spaced, which simplifies evaluation of the policy functions. If unevenly spaced nodes are desired, then `Fallterp*` must be modified to correctly locate the nearest nodes.

A.3 INTEGRATION

A model with both continuous and discrete stochastic variables requires two types of numerical integration. For continuous stochastic variables we apply either the Trapezoid rule or Gauss-Hermite quadrature, and for discrete random variables we use the corresponding transition matrix to weight each outcome by its likelihood.

A.3.1 TRAPEZOID RULE

Suppose there are m realizations of the stochastic component, ε, in the process for some continuous variable z. Since these realizations show up in agents' expectations, we perform numerical integration to average across each of these m realizations. The trapezoid rule is one method of numerical integration. Assuming uniformly spaced realizations of ε, the formula for the trapezoid rule is given by

$$\begin{aligned} E_t[\Phi(\cdot, z_{t+1})] &\approx \frac{\Pr(\varepsilon_1)\Phi(\cdot, z_{t+1}(\varepsilon_1)) + \Pr(\varepsilon_2)(\Phi(\cdot, z_{t+1}(\varepsilon_2)))}{2}\Delta\varepsilon \\ &\quad + \frac{\Pr(\varepsilon_2)\Phi(\cdot, z_{t+1}(\varepsilon_2)) + \Pr(\varepsilon_3)\Phi(\cdot, z_{t+1}(\varepsilon_3))}{2}\Delta\varepsilon \\ &\quad + \frac{\Pr(\varepsilon_{m-1})\Phi(\cdot, z_{t+1}(\varepsilon_{m-1})) + \Pr(\varepsilon_m)\Phi(\cdot, z_{t+1}(\varepsilon_m))}{2}\Delta\varepsilon \\ &= \frac{\Delta\varepsilon}{2}\left[2\sum_{i=1}^{m}\Pr(\varepsilon_i)(\Phi(\cdot, z_{t+1}(\varepsilon_i))) - \Pr(\varepsilon_1)\Phi(\cdot, z_{t+1}(\varepsilon_1)) - \Pr(\varepsilon_m)\Phi(\cdot, z_{t+1}(\varepsilon_m))\right], \end{aligned}$$

where $\Delta\varepsilon$ is the distance between stochastic realizations, $\Pr(\varepsilon_i)$ is the probability of realization i, and Φ is the value of the contents of the expectation operator, given the state of the economy. To obtain the weights (the probabilities) in the trapezoid rule, truncate the distribution of the stochastic variable. For normal random variables, we recommend truncating the distribution at no less than four standard deviations, since omitting more of the distribution often leads to inaccurate results.

A.3.2 GAUSS-HERMITE QUADRATURE

Another commonly employed method of numerical integration is Gauss-Hermite quadrature. Suppose a shock, u, to a continuous variable, z, is normally distributed with mean μ and variance σ^2. Then expectations can be written as

$$E_t[\Phi_{t+1}(\cdot, z_{t+1}(u))] = (2\pi\sigma^2)^{-1/2} \int_{-\infty}^{\infty} \Phi_{t+1}(\cdot, z_{t+1}(u)) e^{-(u-\mu)^2/(2\sigma^2)} du.$$

Applying the change of variables, $\varepsilon = (u-\mu)/(\sqrt{2}\sigma)$, the Gauss-Hermite quadrature rule is

$$\begin{aligned} E_t[\Phi_{t+1}(\cdot, z_{t+1}(u))] &= \pi^{-1/2} \int_{-\infty}^{\infty} \Phi_{t+1}(\cdot, z_{t+1}(\sqrt{2}\sigma\varepsilon + \mu)) e^{-\varepsilon^2} d\varepsilon \\ &\approx \pi^{-1/2} \sum_{i=1}^{n} \omega_i \Phi_{t+1}(\cdot, z_{t+1}(\sqrt{2}\sigma\varepsilon_i + \mu)), \end{aligned}$$

where ε_i are the realizations of the standard normal shock, Φ is the value of the contents of the expectation operator, and ω_i are Gauss-Hermite weights given by $\omega_i = 2^{n+1} n! \sqrt{\pi} [H_{n+1}(\varepsilon_i)]^{-2}$. H_{n+1} is the physicists' Hermite polynomial of order $n+1$.[13]

We usually adopt the Trapezoid rule over Gauss-Hermite quadrature because it is more stable. Moreover, with dense and wide enough grids (at least 10 points and 4 standard deviations) for the continuous shocks, the optimal policy functions under these two methods of numerical integration are virtually identical, even though the Trapezoid rule relies on a truncated distribution.

A.3.3 MARKOV CHAIN INTEGRATION

Suppose a discrete stochastic variable, z, evolves according to an m-state first-order Markov chain.[14] Once again, these realizations show up in agents' expectations, and we must integrate across these m realizations conditional on the previous state. Suppose the transition matrix is given by

$$\begin{aligned} P &= \begin{bmatrix} \Pr[S_t = 1 | S_{t-1} = 1] & \Pr[S_t = 2 | S_{t-1} = 1] & \cdots & \Pr[S_t = m | S_{t-1} = 1] \\ \Pr[S_t = 1 | S_{t-1} = 2] & \Pr[S_t = 2 | S_{t-1} = 2] & \cdots & \Pr[S_t = m | S_{t-1} = 2] \\ \vdots & \vdots & & \vdots \\ \Pr[S_t = 1 | S_{t-1} = m] & \Pr[S_t = 2 | S_{t-1} = m] & \cdots & \Pr[S_t = m | S_{t-1} = m] \end{bmatrix} \\ &= \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{bmatrix}, \end{aligned}$$

where $0 \leq p_{ij} \leq 1$ and $\sum_{j=1}^{m} p_{ij} = 1$ for all $i \in \{1, 2, \ldots, m\}$. Then the conditional expectation can be written as

$$E_t\left[\Phi_{t+1}(\cdot, z_{t+1}) | S_t = i\right] = \begin{bmatrix} p_{i1} & p_{i2} & \cdots & p_{im} \end{bmatrix} \begin{bmatrix} \Phi_{t+1}(\cdot, z_{1,t+1}, S_t = i) \\ \Phi_{t+1}(\cdot, z_{2,t+1}, S_t = i) \\ \vdots \\ \Phi_{t+1}(\cdot, z_{m,t+1}, S_t = i) \end{bmatrix}.$$

[13] We provide a function, `ghquad.m`, to compute the Gauss-Hermite weights. To calculate the coefficients of the Hermite polynomial, `ghquad.m` requires `HermitePoly.m`, which is written by David Terr and readily available on the MATLAB file exchange.

[14] Recall that higher order Markov chains can always be described by a first-order transition matrix.

If a model contains both continuous and discrete stochastic variables, first integrate across the continuous random variables to obtain a set of values, conditional on the realizations of the discrete stochastic variable. Then weight each of these values by their corresponding likelihood. This process yields an expected value across all stochastic components in the model.