Computationally Efficient Solution and Maximum Likelihood Estimation of Nonlinear Rational Expectations Models

by

Jeffrey C. Fuhrer and C. Hoyt Bleakley*

August 10, 1996

Preliminary. Comments Welcome.

Abstract

This paper presents new, computationally efficient algorithms for solution and estimation of nonlinear dynamic rational expectations models. The innovations in the algorithms are as follows: (1) The entire solution path is obtained simultaneously by taking a small number of Newton steps, using analytic derivatives, over the entire path; (2) The terminal conditions for the solution path are derived from the uniqueness and stability conditions from the linearization of the model around the terminus of the solution path; (3) Unit roots are allowed in the model; (4) Very general models with expectational identities and singularities of the type handled by the King-Watson (1995a,b) linear algorithms are also allowed; and (5) Rank-deficient covariance matrices that arise owing to the presence of expectational identities are admissible. Reasonably complex models are solved in less than a second on a Sun Sparc20. This speed improvement makes derivative-based estimation methods feasible. Algorithms for maximum likelihood estimation and sample estimation problems are presented. (JEL E52, E43)

---

* Vice President and Research Assistant, respectively, Research Department, Federal Reserve Bank of Boston, Boston, MA, 02106. The views expressed in this paper are the authors' and do not necessarily reflect those of the Federal Reserve Bank of Boston or the Board of Governors of the Federal Reserve System. Please do not quote without permission of the authors.

For a variety of reasons, nonlinear models for macro- and micro- economics have grown in popularity in recent years. In macroeconomics, the recognition that most linear models cannot capture turning points in business cycles, the inherent nonlinearity in the consumer's budget constraint with time-varying interest rates, the presence of nonlinear adjustment costs in investment, and the nonlinearity of the convex production function all require some accommodation of nonlinearity.

Researchers have employed a number of alternate strategies for computing the solutions to nonlinear models. Their approaches may be separated into three broad categories.

1. Linearize or log-linearize the system as in Kydland and Prescott (1982). In this case, one can apply the techniques developed for linear models.

2. Solve a reduced form version of the system by numerical integration and iteration using dynamic programming or the finite-element method, as in Christiano (1990) and McGrattan (1996), respectively.

3. Numerically solve for the model-consistent path of expectations (in the case of certainty equivalence) from an initial guess, as in Fair and Taylor (1983).

Linearizing models involves approximations that can be evaluated only for simple, analytically tractable cases. Dynamic programming techniques generally require considerably more computing time, often several orders of magnitude greater than linear methods. This paper presents a method that provides a compromise between these two extremes, in the spirit of Fair and Taylor. The method directly solves the nonlinear functions that make up the model. However, it solves a perfect-foresight version of the functions, and thus does not fully incorporate the stochastic features of the model into the solution technique.

The algorithm presented here uses Newton's method to jointly solve for the full time-path of nonlinear equations in the model. It utilizes the sparsity of the system to economize on computations (and storage). The method achieves a computational speed that makes derivative-based estimation methods feasible. The results discussed in section 2.3 suggest that, at least for

some canonical nonlinear models, the omission of the stochastic features of the model is of second- or third-order importance in solution accuracy. How good an approximation the perfect-foresight nonlinear method is for other models remains a question for further research; we discuss some methods for improving the accuracy of the approximation.

The rest of the paper is organized as follows. Section 1 presents the solution algorithm. In section 2, we apply the solution technique to a nonlinear sticky-price model and also to the stochastic growth model. We also compare our solution of the stochastic growth model with results from dynamic programming. Section 3 describes the maximum likelihood estimation procedure using our method, and includes an example. Section 4 suggests several avenues of future research. Section 5 concludes.

1 The Solution Algorithm

In an unpublished paper, Anderson and Moore (1986) sketch a solution algorithm for nonlinear dynamic rational expectations models. This section describes a solution algorithm that is related to the work of Anderson and Moore and to undocumented work by Brian Madigan of the Federal Reserve Board.

Consider a nonlinear rational expectations model characterized by the system of equations

$$F(x_-, x, E(x)) = \epsilon \tag{1}$$

where x is an n-vector of endogenous variables, where dependence of the function on lagged x is summarized in x_-, where $E()$ denotes the expectation operator,[1] and ϵ is an n_e-vector of structural shocks to the system; n is not necessarily equal to n_e, although n_e cannot exceed the number of observed variables in the system.[2]

The solution methods discussed in this paper will be, in essence, perfect-foresight solution methods. That is, the structural shocks are linearly sepa-

[1] The viewpoint date for the expectation operator is considered in detail below.

[2] If the number of structural shocks exceeds the number of observable variables, the Jacobian of transformation will not be full rank.

rable from the nonlinear structure of the economy. The estimation methods discussed in the second half of the paper will maintain this assumption. Other than this restriction, however, the form of the nonlinearities admitted by the technique can be quite general. In this regard, the solution method follows the spirit of Fair and Taylor's (1983) work.

The solution method seeks a sequence of expectations $E_{t-v}(x_{t+j}), j = 1, \ldots, T$ such that, given initial conditions, terminal conditions, and these expectations, the function values for every period in the solution trajectory are zero:

$$F(x_t, E(x_{t+})^* | x_{t-}) = 0 \quad \forall t \, .$$

Now consider a solution path that extends from periods t to T. For any time period s, the Newton step that determines the change in the solution value of x from iteration k to iteration $k+1$ may be written[3]

$$\Delta x_{k+1,s} = J_k^{-1} F(x_k, s) \tag{2}$$

where J_k is the Newton Jacobian matrix, i.e. the matrix of partial derivatives of the functions F with respect to the vector of variables x. However, one can solve for the entire time path $s = t, \ldots T$ of solution vectors at once by writing the stacked set of Newton equations for iteration k as

$$\Delta X_{k+1} = S_k^{-1} F(X_k) \tag{3}$$

where S_k is the Newton-Jacobian matrix for the entire solution path as of iteration k, and X_k is the vector of endogenous variables over the entire solution path. $F(X_k)$ is likewise the vector of function values for the entire solution path, evaluated at the last value of X.

[3]The classical definition of a Newton step for solving a set of nonlinear equations begins with the first-order Taylor expansion of the functions F around the vector x_k

$$F(x_k) = F'(x_k)(x_{k+1} - x_k)$$

and then solves for x_{k+1} by inverting the expansion.

1.1 The Structure of the S_k matrix

Define the maximum lag in the structural model by τ and the maximum lead by θ. We will denote by H_j^s the matrices that contain the derivatives of the set of equations in F with respect to the variables x at lag or lead j evaluated at time period s. Most of the S_k matrix will be composed of H_j^s matrices evaluated at different periods s.

The structure of S_k will then be a sparse band-diagonal matrix, with number of bands equal to $n(\tau + \theta + 1)$:

$$S_k = \begin{bmatrix} H_{-\tau}^t & \dots & H_{-1}^t & H_0^t & H_1^t & \dots & H_\theta^t & 0 & 0 & \dots \\ 0 & H_{-\tau}^{t+1} & \dots & H_{-1}^{t+1} & H_0^{t+1} & H_1^{t+1} & \dots & H_\theta^{t+1} & 0 & \dots \\ & & & & \vdots & & & & & \end{bmatrix} \tag{4}$$

For example, consider the simple set of nonlinear equations

$$\begin{aligned} y_t &= \beta x_{t-1}^2 + (1-\beta) E_t x_{t+1}^2 \\ x_t &= \rho x_{t-1} \end{aligned}$$

Then the H_j^s matrices for $s = t$ and $j = -1, 0, 1$ would be

$$\begin{aligned} H_{-1}^t &= \begin{bmatrix} 0 & 2\beta x_{t-1} \\ 0 & \rho \end{bmatrix} \\ H_0^t &= \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \\ H_1^t &= \begin{bmatrix} 0 & 2(1-\beta) x_{t+1} \\ 0 & 0 \end{bmatrix} \end{aligned}$$

and to form the S_k matrix, they would stack up as in equation 4:

$$S_k = \begin{bmatrix} 0 & 2\beta x_{t-1} & -1 & 0 & 0 & 2(1-\beta)x_{t+1} & 0 & 0 & 0 & \dots \\ 0 & \rho & 0 & -1 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 2\beta x_t & -1 & 0 & 0 & 2(1-\beta)x_{t+2} & 0 & \dots \\ 0 & 0 & 0 & \rho & 0 & -1 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$

1.2 Initial and Terminal Conditions

It is convenient to fold the initial and terminal conditions for the solution into the Newton problem. The initial conditions are simply the lagged data $[x_{t-\tau}, \ldots, x_{t-1}]$. Thus including an $n\tau$ identity matrix in the upper left corner of S_k will assure that these initial conditions hold exactly throughout the solution iterations.[4]

The equations require $n\theta$ terminal conditions for the θ leads that "overhang" the end of the solution trajectory. These conditions are obtained by using the stability conditions for the linearized model to solve out for the leads in terms of previous solution values. The stability conditions come from the method of Anderson and Moore (1985) for linear models. That algorithm computes the solution to a linear rational expectations model of the form

$$\sum_{i=-\tau}^{0} H_i x_{t+i} + \sum_{i=1}^{\theta} H_i E_t(x_{t+i}) = \epsilon_t \,. \tag{5}$$

The algorithm stores the stability conditions in a matrix Q, which satisfies

$$Qx_+ = 0 \tag{6}$$

where $x_+ = [x_{t-\tau+1}, \ldots, x_t, x_{t+1}, \ldots, x_{t+\theta}]$ and Q is a $n\theta$ by $n(\theta+\tau)$ matrix that solves for $[x_{t+1}, \ldots, x_\theta]$ in terms of $[x_{t-\tau}, \ldots, x_t]$. [5] Thus we can write the Newton iteration for the terminal conditions as

$$Q\Delta x_+^k = Qx_+^{k-1} \,. \tag{7}$$

[4]The Newton equation for these conditions is simply $I\Delta x_{t-} = 0$; given starting values for the initial conditions, they will not change over the iterations.

[5]A model linearized about its steady state might no longer tend to that steady state. We therefore modify the linearized model to insure that the linearized terminal conditions imply the correct steady state. To do this, we first compute the steady state, x^*_{t+i}, of the model and compute the H_i matrices. (The stationary variables are set to their steady states and the unit-root variables are set to levels consistent with the stationary variables.) We then adjust the additive constants in the linearized model such that $\sum_{i=-\tau}^{\theta} H_i x^*_{t+i} = 0$. This insures that the linearized model will attain the same steady state as the nonlinear model.

The Newton step for the full set of nonlinear equations, incorporating initial and terminal conditions, is defined by

$$\begin{bmatrix} & I_{n*\tau} & & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ H^t_{-\tau} & \dots & H^t_{-1} & H^t_0 & H^t_1 & \dots & H^t_\theta & 0 & 0 & \dots \\ 0 & H^{t+1}_{-\tau} & \dots & H^{t+1}_{-1} & H^{t+1}_0 & H^{t+1}_1 & \dots & H^{t+1}_\theta & 0 & \dots \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & \dots & & & Q & & & \end{bmatrix} \begin{bmatrix} \Delta x_{t-\tau} \\ \vdots \\ \Delta x_{t-1} \\ \Delta x_t \\ \Delta x_{t+1} \\ \vdots \\ \Delta x_T \\ \Delta x_{T+1} \\ \vdots \\ \Delta x_{T+\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ F(x_t) \\ F(x_{t+1}) \\ \vdots \\ F(x_T) \\ Qx_+ \end{bmatrix} . \tag{8}$$

1.3 Iteration

For initial guesses of the solution vector X^0, one can compute the function values $F(X^0)$, the derivative matrices H^s_j that compose the main body of S_k, and the stability conditions Q that determine the initial conditions. Solution of the Newton step equation 8 yields a new value of X. The derivative matrices, stability conditions, and function values are updated, and the process is iterated until numerical convergence is achieved.[6] For most problems, the convergence criterion is that the maximum of the absolute value of the function values not exceed a critical limit.

1.4 Computational Efficiency Considerations

Three elements of the solution algorithm provide substantial gains in computational efficiency:

[6] A Newton step might place you outside the domain of the function being solved. In such cases, we implemented a line-search algorithm along the direction of the Newton step.

1. The derivative matrices H_j^s are computed analytically, rather than numerically, so that fast functions can be called to evaluate them at each solution iterate, rather than computing them numerically at each iterate;

2. The algorithm keeps track of time-varying and constant elements of the derivative matrices, evaluating only the time-dependent elements at each iteration;

3. The sparsity of the S_k matrix is taken into account, so that the speed of solution of the sparse linear Newton equation is improved by orders of magnitude.

The last improvement alone increases the speed for each iteration by more than two orders of magnitude for a model with 10 equations and solution path of length 50.[7]

2 Examples and Benchmarking

In this section, we apply the solution technique to two models with nonlinear structure. The first model, discussed in 2.1, contains a nonlinear term structure equation and a boundary at zero on the nominal interest rate in the monetary policy reaction function. Next, in 2.2, we present two stochastic growth models which are substantially "more" non-linear. When evaluating the effect of a single shock and using the steady state as the initial solution path, these models solve very quickly (in under one second) on a Sun Sparc20.

[7] We have coded the nonlinear solution algorithm in Matlab. In addition, we have developed a modeling language in which one can express a general nonlinear rational expectations model. Our software parses the model, takes analytic derivatives of the model equations, and writes out Matlab functions that evaluate the derivatives and equation residuals. We have also developed software for likelihood function evaluation and maximum likelihood estimation (described later), which we link to one of Matlab's optimizers to find the maximum of the constrained likelihood.

In all cases, we define the convergence criterion to be as above, i.e. $max(abs(F)) \leq tol$, and a convergence tolerance of 1×10^{-6} ensures that we find very near-zeros of all the functions.

2.1 A Nonlinear Sticky-Price Model

We use a simple sticky-price macro model to illustrate the solution algorithm. The model is nearly identical to the model used in Fuhrer and Madigan (1994), and for a full description, the reader is referred to that paper.

The two equations with nonlinearities include the real term structure equation that equates the *ex ante* holding period return to a real consol bond paying real interest rate ρ_t to the real holding period return on a short-term bond, which is just the nominal coupon i_t less expected inflation π_{t+1}

$$\rho_t - \frac{1}{\rho_{t+1}}[\rho_{t+1} - \rho_t] = i_t - \pi_{t+1}, \tag{9}$$

where π_t is the backward first difference of the log price level. A conventional linear approximation to this equation is the constant-duration approximation that reduces equation 9 to a first-order linear equation.

The (economically) more important nonlinearity arises in the equation that defines the policy response of the federal funds rate to deviations of the monetary authority's ultimate goal variables from their targets. A simple way of imposing a non-negativity constraint on the nominal interest rate is to write the policy reaction function as

$$log(i_t) - log(i_{t-1}) = \alpha_\pi(\pi_t - \pi^*) + \alpha_y(y_t - y^*) + \epsilon_{it}\,. \tag{10}$$

The remaining equations, summarized below, are linear and describe the behavior of sticky prices (due to contracting), aggregate demand, and potential output.

$$\begin{aligned} \tilde{y}_t &= \alpha_{y_1}\tilde{y}_{t-1} + \alpha_{y_2}\tilde{y}_{t-2} - \alpha_\rho(\rho_{t-1} - \bar{\rho}) + \epsilon_{yt} \qquad (11) \\ p_t &= \sum_{i=0}^{3} f_i x_{t-i} \end{aligned}$$

$$
\begin{aligned}
v_t &= \sum_{i=0}^{3} f_i (x_{t-i} - p_{t-i}) \\
x_t - p_t &= \sum_{i=0}^{3} f_i E_t (v_{t+i} + \gamma \tilde{y}_{t+i}) + \epsilon xt \\
&= \sum_{i=1}^{3} \beta_i (x_{t-i} - p_{t-i}) + \sum_{i=1}^{3} \beta_i E_t (x_{t+i} - p_{t+i}) + \gamma^* \sum_{i=0}^{3} f_i E_t (\tilde{y}_{t+i}) + \epsilon xt
\end{aligned}
$$

The first equation defines the aggregate demand curve, which makes the output gap $\tilde{y}_t$ a function of two of its own lags and of the lagged long-term real interest rate. The second equation defines the price index p_t as the weighted sum of current and past contract prices x_{t-i}, where the weight on contract prices previously negotiated and still in effect is denoted f_i, and $f_i = .25 + (1.5 - i)\, s, 0 < s \leq 1/6, i = 0, \ldots, 3$. The third defines the real contract price index v_t as a weighted average of current and past real contract prices. The last two equations express the fundamental contracting equation of the model presented in Fuhrer and Moore (1995b). The convoluted weights β_i and γ^* in the last two equations are defined as

$$
\begin{aligned}
\beta_i &= \sum_{j=0}^{3} \frac{f_j f_{i+j}}{1 - \sum_{j=0}^{3} f_j^2} \\
\gamma^* &= \frac{\gamma}{1 - \sum_{j=0}^{3} f_j^2}
\end{aligned}
\tag{12}
$$

The number of equations for this model is 7, the maximum lag is 3, and the maximum lead is 3. The model has a well-defined steady state characterized by inflation at its target ($\pi = \pi^*$), output equal to potential, real rates at their long-run equilibrium, and nominal rates obeying the Fisher identity.

To solve this model, we set the initial conditions to a 4 percent inflation steady state.[8] The symbolic derivatives of the model equations with respect

[8]Note that the variables that define the price index and the contract price have unit roots; the steady-state initial conditions have all of these nominal levels growing at the inflation rate.

to the endogenous variables are computed,[9] and functions are written to compute the time-varying and non-time-varying derivatives given the current values of the endogenous variables and the parameters. The steady-state inflation parameter in the model is set to 0, so that the simulation will depict the effects of an unanticipated disinflation from 4 to 0 percent. For this simulation, we assign the remaining parameters of the model as follows:

Parameter	Value
IS Curve	
α_{y_1}	1.25
α_{y_2}	-.42
α_ρ	.3
$\bar{\rho}$	.03
Reaction Function	
α_π	1
α_y	1.5
Contracting	
s	.08
γ	.005

The initial solution path is computed from the model linearized about the new steady-state values. For a solution horizon of 50 periods, the solution converges in three Newton steps, taking a total of 1.1 seconds. Figure 1 displays the solution trajectories for the disinflation simulation. As the figure illustrates, the variables are quite near their new steady-state values at the end of the solution trajectory, so the first-order approximation to the model at that point should be good, and the stability conditions that pin down the terminal conditions should be quite accurate for the model.

2.2 Two Stochastic Growth Models

The first growth model that we examine is the simple single-input stochastic growth model studied in Taylor and Uhlig (1990). The concave production

[9] The Maple kernel, called from Matlab, performs these symbolic operations.

function f employs end-of-period capital, k_{t-1}, augmented by the technology level, z_t, to produce current output, y_t:

$$y_t \equiv f(k_{t-1}, z_t) = z_t \, k_{t-1}^{\alpha} \, . \tag{13}$$

The log of the technology is assumed to be first-order autoregressive

$$\ln(z_t) = \rho \, \ln(z_{t-1}) + \varepsilon_t \, . \tag{14}$$

Current-period utility is isoelastic, $u_t = c_t^{1-\gamma}/(1-\gamma)$, and total utility is time-separable and discounted at a rate β:

$$U_t = \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma} \, . \tag{15}$$

Total income comprises the sum of consumption and net investment, yielding the familiar identity

$$y_t = c_t + k_t - (1-\delta) \, k_{t-1} \, . \tag{16}$$

By taking the derivative of the utility function with respect to k_t, we derive the also-familiar first-order condition:

$$U_{c_t} = \beta U_{c_{t+1}} \left(f_{k,t+1} + 1 - \delta \right) \, .$$

When ρ is less than one in absolute value, the model implies a stationary steady state for the two state variables k_t and z_t. Evaluating equations 14, 13, and 16 in the steady state, we obtain steady-state solutions for the technology shock, capital, and consumption:

$$\begin{aligned} z^* &= 1 \, , \\ k^* &= \left(\frac{1/\beta - (1-\delta)}{\alpha \, z^*} \right)^{\frac{1}{\alpha - 1}} \, , \\ c^* &= z^* \, k^{*\alpha} - \delta k^* \, . \end{aligned}$$

To compute a solution path for the model, we set the parameters as shown in the table below, shock the technology by 10 percent, and use the steady state for the model as the initial solution path. We compute a 50-period solution path for the model.

Parameter	β	α	γ	δ	ρ
Value	0.99	0.33	0.5	0.1	0.9

The solution path converges in 3 Newton steps with the largest function deviation 1.8×10^{-11} in 0.17 seconds. The solution paths for capital, technology, and consumption are displayed in figure 2.

The second model is the canonical RBC model examined in McCallum (1989). A labor–leisure choice is now included, and utility thus depends on consumption and leisure. The income identity and technology shock evolve as in equations 16 and 14 above. Output is produced according to a two-factor Cobb-Douglas production function:

$$y_t \equiv f(k_{t-1}, n_t, z_t) = z_t \, k_{t-1}^{\alpha} \, n_t^{1-\alpha} \, . \tag{17}$$

Period t utility depends on consumption, c_t, and leisure, defined as $1 - n_t$; the following Cobb-Douglas utility function is assumed

$$u_t = c_t^{\gamma} \, (1 - n_t)^{1-\gamma} \ . \tag{18}$$

Taking derivatives, we construct first-order conditions with respect to capital and labor:

$$\begin{aligned} u_{c,t} &= \beta u_{c,t+1} \left(f_{k,t+1} + 1 - \delta \right) , \\ u_{n,t} &= u_{c,t} \, f_{n,t} \, . \end{aligned} \tag{19}$$

Because we chose Cobb-Douglas functional forms, the first derivatives of the utility and production functions can be conveniently written:

$$\begin{aligned} u_{c,t} &= \frac{(1-\gamma) u_t}{c_t} \\ u_{n,t} &= \frac{-(1-\gamma) u_t}{(1-n_t)} \\ f_{k,t+1} &= \frac{\alpha y_t}{k_{t+1}} \\ f_{n,t} &= \frac{(1-\alpha) y_t}{n_t} \end{aligned} \tag{20}$$

Again, for values of ρ less than one in absolute value, the model implies a stationary steady state. The analytic solutions for the steady-state values of the state variables are

$$\begin{aligned} z^* &= 1 \\ n^* &= \frac{\gamma(1-\alpha)}{1-\gamma\alpha+\frac{\alpha\beta\delta(\gamma-1)}{1-\beta(1-\delta)}} \\ k^* &= n^* \frac{1/\beta-(1-\delta)}{\alpha}^{\frac{1}{\alpha-1}} \end{aligned} \tag{21}$$

We compute a 50-period solution path for the model, setting the parameters in the model as shown in the table below, shocking technology by 10 percent, and using the steady state for the model as the initial solution path.

Parameter	β	α	γ	δ	ρ
Value	0.99	0.33	0.5	0.1	0.9

The path converges in 3 Newton steps with a maximum function deviation of 2.2×10^{-10} in 0.44 seconds. A 20-period solution solves in 3 Newton steps to the same accuracy in 0.18 seconds. The solution paths for capital, labor, technology, and consumption are displayed in figure 3.

2.3 Comparison with Results from Dynamic Programming

We compare our results with similar ones summarized in Taylor and Uhlig (1990) and a simple dynamic programming (DP) exercise. We use as a benchmark the simple stochastic growth model with capital and consumption trade-off only. This model is presented in detail at the beginning of Section 2.2.

Differences between our results and the previous work using dynamic programming come from two sources:

1. Dynamic programming methods use a finite-sized grid as an approximation to the state-space, whereas our method works with continuous variables (within machine precision).

2. Our method does not account for the effect of the distribution of future shocks on today's expectations, whereas dynamic programming explicitly (albeit approximately) incorporates this into the value function iteration.

We will address these differences each in turn.

To test the error introduced by the discretization of the state space, we compare deterministic solution paths generated by both methods, for a large shock to technology. Our solution is as described in section 2.2.

Our application of dynamic programming to the stochastic growth model follows Christiano (1990). The standard value function iteration is

$$V(k_{t-1}, z_t) = \max_{\{k_t \ni c_t \geq 0\}} \left(u(c(k_{t-1}, k_t, z_t)) + \beta E_t V(k_t, z_{t+1}) \right), \tag{22}$$

where consumption, c_t, is implicitly defined by the budget constraint, and the state-space variables, k and z, are discretized onto a grid. In this exercise the technology level follows a deterministic path and it reverts to its steady state, i.e. $\ln(z_{t+1}) = \rho \ln(z_t)$ exactly. We take advantage of this property and set the grid for z to be the path followed for 101 periods starting from $z_0 = 1.6$. The capital grid is set up such that $k \in [3, 30]$. The grid is divided into N elements each of spacing $27/N$. Therefore, the value function iteration becomes

$$V(k_{t-1}, z_t) = \max_{\{k_t \ni c_t \geq 0\}} \left(u(c(k_{t-1}, k_t, z_t)) + \beta V(k_t, z_t^{\rho}) \right). \tag{23}$$

We iterate the value function equation for different values of N until the value function changes less than 10^{-7} per iteration. The last period, T, of the path is assumed to be close enough to the steady state that we iterate it upon itself, e.g.

$$V(k_T, z_T) = max_{k_T}(u(c_T) + \beta V(k_T, z_T)).$$

Following case 1 in Taylor and Uhlig, the depreciation rate is zero. The parameters and steady state of the model are as follows:

Parameter	β	α	γ	δ	ρ
Value	0.95	0.33	0.5	0	0.9

The solution of the stochastic growth model using DP is asymptotically equivalent to our method. As the grid becomes finer, the DP solution matches our solution more closely. In figure 4, we display solution paths from our method and two DP exercises: one with a capital grid with $N = 100$ elements, and another with $N = 500$ elements. The solution path from DP closely matches the path from our method, when the DP grid has enough elements.

This convergence is also illustrated in Table 1, which displays the root-mean-squared difference between our solution path and the DP solution for various N. As we make the DP grid finer, the solution path deviates less from our method.

Table 1

Root Mean Square Difference
Between Solution Paths from
Dynamic Programming and Nonlinear AIM

DP grid size	75	100	200	300	500
RMS Diff.	0.59	0.45	0.21	0.13	0.07

The computation of the deterministic solution path using DP requires hours, whereas our method requires a few tenths of a second. Of course, once the decision rules are computed for a given set of parameters, generating a new solution path is computationally trivial. However, if we are also interested in estimating parameters, recomputing decision rules by value-function iteration becomes very time-consuming. Therefore, for deterministic solution of the stochastic growth model, our method computes the solution path more accurately and in much less time than dynamic programming.

We now compare the two solution techniques for a stochastic simulation. Here we expect weaker correspondence because our method does not account for the distribution of future shocks in the formation of expectations. For the stochastic growth model, our method assumes instead that $E_t(u(c_{t+1})) = u(E_t(c_{t+1}))$, an assumption which is plainly false for concave

utility and non-zero shock variance. However, in this case this assumption makes little apparent difference for the decision rules.

To model the stochastic growth model when it is genuinely stochastic, we make one additional assumption. Recall the equation governing the technology level in this model: $\ln(z_t) = \rho \ln(z_{t-1}) + \varepsilon_t$. Following case 2 in Taylor and Uhlig, we simply assume that $\varepsilon_t \sim N(0, \sigma_\varepsilon)$. This assumption changes nothing about our solution, but changes the way expectations are computed in the value function iteration for DP.

In computing the decision rules we use the grid of initial conditions from Taylor and Uhlig:

$$\begin{aligned} k_{t-1} &\in \{5, 10, 15, 20, 25\} \\ z_t &\in \{.4, .7, 1, 1.3, 1.6\} \end{aligned}$$

We estimate decision rules from our method by solving for the solution paths for each initial condition. As a benchmark, the dynamic programming results presented by McGrattan (1996) are used. Parameter settings correspond to Taylor and Uhlig's "case 2," i.e.

Parameter	β	α	γ	δ	ρ	σ_ε
Value	0.95	0.33	1.5	0.1	0.95	.1

It is of note is that σ_ε implies that the technology shock will exceed 10 percent in absolute value about one third of the time. This represents an implausibly large variance for technology.

Despite the noisy technology, consumption decision rules from both methods are nearly identical. The computed consumption choice differs by less than 1 percent at every point in the grid. Table 2 presents these results.

In each case, DP computes decision rules for consumption that are slightly lower than those computed by our method. These results conform to a rough intuition about this simple model. One might expect the optimal consumption path under uncertainty to lie below the path under certainty equivalence due to a precautionary savings motive. As predicted, the decision rules for consumption differ for this rather noisy technology; however, the differences are quite small.

Table 2

Decision Rules for Consumption
Computed by
Dynamic Programming and Our Method

	z_t				
k_{t-1}	.40	.70	1.00	1.30	1.60
	Our Method				
5	0.86	1.12	1.35	1.58	1.79
10	1.33	1.65	1.94	2.22	2.48
15	1.73	2.10	2.43	2.74	3.04
20	2.09	2.50	2.87	3.21	3.53
25	2.44	2.88	3.27	3.64	3.98
	Dynamic Programming				
5	0.86	1.11	1.34	1.57	1.78
10	1.32	1.64	1.93	2.20	2.46
15	1.71	2.09	2.41	2.72	3.01
20	2.08	2.49	2.85	3.19	3.50
25	2.42	2.86	3.25	3.61	3.95

McGrattan (1996) also presents consumption decision rules computed using the so-called finite-element method (FEM). These consumption decisions correspond to those computed by our method. Figure 5 displays the consumption decision rules computed by both methods. The decision rules for the FEM method lie extremely close to those obtained using our method. [10]

Taking these comparisons with two conventional solution methods together, we conclude that our solution method as applied to the standard

[10] Perhaps of note is that the FEM consumption decisions are not always lower than those from our method. This is somewhat odd given that the FEM approximation incorporates information about the distribution of future shocks, whereas our method represents the case of certainty equivalence. As noted above, one might therefore expect the FEM solution for consumption to be lower than ours at every point.

stochastic growth models is fast (orders of magnitude faster), accurate, and straightforward. Ignoring the effect of the distribution of errors on expectations appears to be an acceptable approximation in these cases.

3 Maximum Likelihood Estimation

Using our solution method, maximum likelihood estimation for many nonlinear models is now computationally feasible. For this purpose, greater precision is required in the description of the nonlinear equations. Several distinctions are of importance:

1. The expectations viewpoint date is now important;
2. The variables that are not data, i.e. that are defined by expectational and other identities, must be distinguished from those variables that are data;
3. The equations that are stochastic, i.e. are shocked by a random error term, must be distinguished from those that are not.

Thus it will be useful to express one nonlinear equation in the system as

$$f(x_-, x^D, x^N, E_k(x_+)) = \epsilon \tag{24}$$

where the vector of current data variables is denoted x_D, the vector of not-data variables by x_N, the lagged data (initial conditions) are denoted x_-, and the expectations viewpoint date by the subscript k attached to the expectations operator, E_k.

The computation of the likelihood value is detailed in the next subsection. The following subsections describe the constraints imposed upon the likelihood and the procedure for maximizing the likelihood.

3.1 Evaluating the Likelihood Function

Following the standard derivation for simultaneous equations in, for example, Amemiya (1983), we write the likelihood as two components: the (determinant of the) residual covariance matrix, and the Jacobian of transformation

from the residuals to the observable variables. Both retrieving the residuals that define the covariance matrix and computing the Jacobian of transformation involve the computation of solution paths for the expectations in the model, which is described above.

Note that in some earlier *linear* rational expectations model estimation, notably Sargent (1978), the restricted reduced-form was estimated directly, thus bypassing the need to compute a Jacobian of transformation. In models that include non-trivial identities, this simplification is not possible, as the residual variance-covariance matrix for the reduced-form model is not full rank. Here, obtaining a closed-form solution for the reduced-form model is not possible in general. Even in cases where it can be obtained, however, when the model includes identities that render the variance-covariance matrix singular, the "structural" approach must be taken.

3.1.1 Retrieving the structural residuals

The $t-1$-period expectation viewpoint case. The procedure comprises three steps: (1) Solve for the path of expectations for all the variables in the model, from periods $t+1$ to the maximum horizon, given information up to and including period $t-1$; (2) Given the initial conditions, the period-t data, and the expectations, compute values of the not-data variables that are consistent with the function definitions; (3) Given initial conditions, expectations, period-t data and period-t not-data variables, compute the residuals of the functions, i.e.,

$$\epsilon_t = f(x_-, x_t^D, x_t^N, x^e)\,. \tag{25}$$

Step (1) is just the computation of a solution path, as described above, with initial conditions for periods $t-\tau$ to $t-1$. Step (2) solves the functions for the values of the not-data variables x_t^N that are consistent with the $t-$period data and the expectations. Step (3) computes the residuals by evaluating the functions at the solved values for x^N and x^e.

The $t-$period expectation viewpoint case. In the $t-1$-period viewpoint case, the determination of expectations and the determination of not-data variables can be separated into two steps, because the expectations are

formed (by definition) without regard to the $t-$period realizations of the not-data (or the data) variables. In the $t-$period case, this cannot be so. Now expectations must be consistent with data up to and including period t, which must include solutions for the $t-$period not-data variables.

As a result, a relatively small modification in the algorithm is required. The solutions for the path of expectations and for the not-data variables are computed simultaneously. In practice, for every iteration in the expectations solution step, the not-data variables that are consistent with that step's expectations, lagged data, and current data are recomputed. The process is considered to converge when both expectations and not-data variables have converged.

3.1.2 Computing the Jacobian

The Jacobian of transformation from residuals to observables, J_t, is simply

$$J_t = \frac{\partial \epsilon_t}{\partial x_t^D} \,. \tag{26}$$

We know of no way to compute this Jacobian analytically, so the derivatives are computed numerically, using two-sided finite differences to approximate the analytical derivative. Each column of the Jacobian matrix is computed by perturbing one data variable in period t, and recomputing the vector of residuals that arises. In the $t-1$-period viewpoint case, this simply involves re-solving for the not-data variables and recomputing the residuals, since the expectations are not altered when a $t-$period variable is perturbed. In the $t-$period viewpoint case, the expectations must be recomputed for a perturbed $t-$period data value, and the residuals recomputed, to obtain the derivative estimate.

In general, the Jacobian will vary over time, so that the Jacobian must be computed for each period in the sample. In practice, for $t-$period expectations this would increase computation time by a factor equal to twice the number of data variables, since each computation would entail computing the residuals for the covariance matrix *and* twice again for each derivative required for the Jacobian. As implemented, the algorithm computes the Jacobian at the beginning and end of the sample and averages those estimates.

When the Jacobian does not vary too much over the sample, this approximation will do little harm to the precision of the likelihood function.[11]

3.2 The Likelihood Function

Once both residual covariance matrix and Jacobian are computed, the concentrated log-likelihood is defined conventionally as

$$\pounds = T \log(\|J\|) - .5 * \log(|\Omega|)\,. \tag{27}$$

3.3 Numerical Maximization of the Log-Likelihood

In the discussion above, the dependence of the functions f on the parameters of the model has been suppressed for simplicity. Expressing the likelihood function's dependence on the parameter set β,

$$\pounds = \pounds(x, \beta)\,. \tag{28}$$

The task of the maximum likelihood estimator is to find values of β that maximize equation (28). To do so, we use a sequential quadratic programming algorithm from the Matlab optimization toolbox. As implemented, the algorithm uses numerical two-sided derivatives for both the likelihood gradient and the constraint gradient.

Practical considerations. The evaluation of the likelihood entails many solutions of the model, and therefore we endeavour to reduce the time required for solution. An initial guess for the solution path should be as "solvable" as possible. To this end, we adopted several strategies. In many cases, initializing the solution path with the steady state of the model was adequate. Some time savings were realized by using an initial path computed with the linearized model.[12] For the model in the estimation example, it is

[11] Fair (1984) successfully implements this Jacobian approximation method. For the test example described below, recomputing the Jacobian for every period had a trivial effect on the value of the likelihood function.

[12] See Anderson and Moore (1985) for details on this method or Fuhrer and Moore (1995a) for a detailed application.

necessary to use the linearization about the steady state when initializing the path and computing the terminal conditions, although this is not generally true.

Since the likelihood evaluation consists of many solution paths generated sequentially from the data, we use information from the $(T-i)$th period's expectations to aid in efficiently computing the solution for period $(T-i+1)$. We use the converged expectations path from $T-i$ as an initial guess for $T-i+1$, in the hope that the expectations are not changed enormously by the information update. We also reduce the length of the solution path (shorten the horizon) if the system moves close to the steady state well before end of the horizon in the previous period.

Finally, a solution path may be difficult to find in some regions of the parameter space. Therefore, we have found that limits on the parameters are often helpful for the initial estimations.

3.3.1 Constraints Imposed on the Maximization Problem

In implementing a maximum likelihood estimator for linear problems, Fuhrer and Moore impose a "root constraint" on the estimator.[13] For the linear model, the solution will not be unique and stable unless the parameters imply the correct number of roots of modulus greater than one. For the nonlinear model, this condition should hold in a linearization about the steady-state. In practice, we impose the root condition for the linearization of the model about the endpoint of the expectation solution path.

The total number of required stability conditions for a model with n equations and θ leads is $n\theta$. In general, a set of "auxiliary" stability conditions that arise from singularities in the lead blocks of the equations will provide some number n_s of the required stability conditions.[14] The remainder come from the spanning vectors (in most cases the eigenvectors) associated with the large roots in the model. For given parameter values, one can compute the number of auxiliary stability conditions n_s. For the same parameters, one can compute the roots of the transition matrix for the model, and sort

[13] See Fuhrer and Moore (1995b) for an application of this estimation strategy.

[14] See Anderson and Moore (1985) and King and Watson (1995a) for details.

them from small to large. The location of the smallest large root will be $n\theta - n_s$, and the location of the largest small root will be one more than this. We set the numerical definition of a large root to be a number slightly larger than one, in most cases $1 + 1 \times 10^{-6}$. Denoting the upper bound by U, the numerical tolerance for the constraint by t, and the vector of the modulus of the roots by λ, the constraint may be expressed as

$$\min(U - \lambda(n\theta - n_s r + 1), \lambda(n\theta - n_s r) - U - t) > 0\,. \tag{29}$$

This ensures that at the expected break point, the smallest large root is t greater than the upper bound U and that the largest small root is less than or equal to U.

The augmented Lagrangian optimization method allows the constraint to be violated during optimization; the constraint Jacobian always points the solution vector back in the direction of the feasible range, and a converged solution always satisfies the constraint.

3.3.2 Standard Errors

The standard errors for the estimated parameters may be obtained from an estimate of the inverse Hessian. The standard errors may be computed as

$$\left(\sqrt{-\hat{H}^{-1}}\right)_{i,i}\,,$$

the square-root of the diagonal elements of the negative of the inverse Hessian matrix. The optimization algorithm that we employ returns a Hessian estimate based on the BFGS updating algorithm (see Gill, Murray, and Wright (1981)). In practice, a more accurate estimate of the local curvature of the likelihood surface may be obtained from direct numerical second derivatives of the likelihood function.

3.4 A Maximum-Likelihood Estimation Example

While models of the type explored in the solution section can be solved quite rapidly, computing the likelihood for this class of models is solution-intensive.

Each evaluation of the likelihood requires at least N solutions, where N is the number of observations in the sample.

We estimate the parameters $[\alpha_\pi, \alpha_y, \bar{\pi}, \alpha_\rho, \bar{\rho}, \gamma]$ of the model described in section 2.1 on a sample from 1965:QI to 1990:QIV. We use as starting values the estimates from a linear version of the model that employs the constant-duration approximation for the term structure equation 9,

$$\rho_t - D[\rho_{t+1} - \rho_t] = i_t - \pi_{t+1} \, ,$$

where D is Macaulay's (1938) duration. The data are as described in Fuhrer and Moore (1995b).[15]

The routine converges after 71 steps and 805 function evaluations with a feasible solution. The maximum-likelihood estimates of the parameters are presented in Table 3. The improvement in the likelihood is extremely significant: The likelihood ratio test for the restriction that holds the parameters at their initial values takes the value 134.44, a χ^2 random variable with 6 degrees of freedom with a p-value of 10^{-24}.[16]

Estimating a potentially misspecified model on real data is a high hurdle for any estimation technique to clear. A somewhat easier task is to estimate the parameters for a data set for which we know the true model. Thus, we generate a simulated data set using the model of this section and the initial parameter values displayed in column 1 of the above table. We then run the FIML procedure on this data set. We initialize the parameters at some distance from their "true" values. The converged values are presented in Table 4 below.[17] The parameter estimates from this exercise generally

[15] We set $\alpha_{y_1} = 1.25$ and $\alpha_{y_2} = -.42$, which are estimated for the linear version of the model by Fuhrer and Moore (1995b).

[16] Grid searches, Monte Carlo parameter selection, and genetic methods yield similar results for the example described above. However, the likelihood function examined is apparently smooth enough that derivative-based search methods are most efficient.

[17] We create a series of shock vectors, $\epsilon_t = [\epsilon_{it}, \epsilon_{yt}, \epsilon_{xt}]^T$. These shocks are normally distributed with the following variance-covariance matrix:

$$\Omega = \begin{bmatrix} 0.7589 & 0.2652 & -0.0589 \\ 0.2652 & 0.8096 & -0.1367 \\ -0.0589 & -0.1367 & 0.2030 \end{bmatrix} \times 10^{-4}$$

Table 3

FIML Estimation Results,
U.S. Quarterly Data 1965:QI–1990:QIV

Parameter	Initial Value	Lower Bound	Upper Bound	Converged Value	Standard Errors	t-stat, $(H_0 : p = 0)$
α_π	0.5	0.001	5	0.303	0.238	1.27
α_y	0.5	0.005	0.2	0.679	0.378	1.80
$\bar{\pi}$	0.03	0.001	5	0.0412	0.0307	1.34
α_ρ	0.79	0.001	1	0.202	0.0925	2.18
$\bar{\rho}$	0.032	0.005	0.2	0.0559	0.0203	2.75
γ	0.00455	10^{-7}	0.3	0.00139	0.00116	1.20
Likelihood value:	1475.65			1542.87		

lie close to the true values. The estimated standard errors imply that only for the α_π parameter could we reject the true value, as indicated in the last column of the table. Thus, for this nontrivial model and data set, the estimator performs quite well.

4 Future Studies

We suggest further research into the approximation error introduced by our method's treatment of expectations. In section 2.3, we show that, for the stochastic growth model with noisy technology, our results closely match

This is the variance-covariance structure of shocks in the linear version of this model, estimated by Fuhrer and Moore (1995b).

We simulate the model with these shocks, using the $t-1$-period viewpoint date for expectations. (See Appendix A.) So that the initial conditions do not have undue influence on the estimate, and the "faux" sample begins 100 periods after the initial period.

Table 4

FIML Estimation Results,
Simulated Quarterly Data 1965:QI–1990:QIV

Parameter	"Truth" (p^*)	Initial Value	Lower Bound	Upper Bound	Converged Value	Standard Errors	t-stat, ($H_0 : p = p^*$)
α_π	0.5	1	0.001	5	0.432	0.034	-1.97
α_y	0.5	1.5	0.005	0.2	0.503	0.038	0.08
$\bar{\pi}$	0.03	0.04	0.001	6	0.0298	0.0021	-0.11
α_ρ	0.79	0.3	0.001	1	0.361	0.233	-1.83
$\bar{\rho}$	0.032	0.04	0.005	0.2	0.0358	0.0030	1.16
γ	0.0045	0.005	0.001	0.3	0.0104	0.00722	0.81
Likelihood value:	1839.1	1713.35			1844.26		

those from dynamic programming despite the fact that we use a certainty equivalence approximation. For some models, this approximation may not be so accurate. In these cases it will be necessary to take into account the distribution of future errors in forming model expectations. We now discuss several proposed methods for doing this. Each method involves applying a certain amount of "brute force" to the problem because we must numerically integrate over the probability density of future shocks.

The most brutish of the brute force methods would be to use our method as a starting point for dynamic programming. In particular, value and decision functions computed by our method would be the initial guess for the value function equation. Presumably the certainty equivalence solution will be a decent first approximation and the number of value function iterations required for convergence will be low.

In looking for an alternative to dynamic programming methods, we note that Fair and Taylor (1983) propose a Monte Carlo integration method

for evaluating expectations with stochastic errors using an extended path method. In particular, many sets of expectations are computed using random draws from the shock distribution over the entire path. These conditional expectations are then averaged to form the unconditional expectation. In effect, this method numerically integrates over the probability distribution of shocks for the entire extended path.

We view this as a backhanded way to evaluate the multiple integrals over the distribution of shocks. An important aspect of the formation of expectations is that a given shock in period $t+i$ could be followed by many different shocks in $t+i+1$. Each of these could be followed by many different shocks in $t+i+2$, and so on. For discrete shocks, these possibilities resemble the branches of a tree. This simulation method, however, takes a single set of shock realizations each time and solves for the expectations conditional on those realizations. The branching phenomenon will only be represented with a very large number of random shocks. While this method asymptotically approximates the unconditional expectation, it is not clear to us that it would converge very quickly.

Instead, we propose a "hybrid" method which would approximate the unconditional expectations without completely sacrificing computational efficiency. We would discretize the shocks into k possible realizations for n periods ahead. All remaining shocks would be set equal to their expected value (presumably zero). For a specific combination of shock realizations, we would solve for the conditional expectations. This last step would be repeated until we have integrated over all the possible combinations of discrete shocks. For more accuracy, one could expand the number of levels of the nested integral. However, in order to numerically integrate over k possible shock values per period in n periods, we would need to generate k^n solutions.

It remains to be seen which of these methods are best suited to various nonlinear models. Future research could better articulate the trade-off between the computational efficiency of the solution method and the numerical accuracy of the unconditional expectations. In the meantime, our solution method can be safely applied to models for which substantial accuracy is not lost because of the certainty equivalence approximation.

5 Conclusion

This paper has developed computationally efficient algorithms for solution and estimation of nonlinear dynamic rational expectations models. The algorithms yield accurate solutions for the models studied here, even when compared to dynamic programming techniques that explicitly account for the distribution of future shocks. The algorithm obtains solutions rapidly, in part by taking advantage of significant sparsity in the structure of the fundamental Newton step. The algorithm also accommodates models with unit roots and with singularities of the type addressed for linear models in Anderson and Moore (1985) and King and Watson (1995a,b).

An important advantage of the rapid solution algorithm is that it makes derivative-based estimation methods feasible. A single computation of the likelihood function might take hours using existing methods; with our algorithm a likelihood evaluation takes on the order of one minute for moderate-sized models and data sets. This allows a researcher to estimate a nonlinear rational expectations model via maximum-likelihood estimation in hours, rather than days or more. Our hope is that this will allow more extensive empirical analyses of many classes of nonlinear rational expectations models.

Appendix A: Stochastic Dynamic Simulation

In section 3.4, we use stochastic simulations to generate "faux" data to test the estimator. This appendix describes the computations required for a stochastic simulation using our solution method. The simulation method differs somewhat depending upon the viewpoint date of the expectations.

We perform the simulation sequentially, shocking and solving for the t-period, incorporating the solution for x_t into the initial conditions for the $(t+1)$-period, and so on. In the case of the t-period viewpoint date, our method forms expectations based on the x's up to $t-1$ and conditional on ϵ_t. (The realizations of x_t and ϵ_t are, of course, not independent.) This is functionally equivalent to taking the conditional expectations using the $t-1$-period information and given ϵ_t. In particular, instead of solving for the zeros of $F(x_{t-}, x_t, E(x_{t+}))$, we find x_t and $E(x_{t+})$ that satisfy $F(x_{t-}, x_t, E(x_{t+})) = \epsilon_t$. Therefore the full Newton step is now

$$\begin{bmatrix} & I_{n*\tau} & & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ H^t_{-\tau} & \dots & H^t_{-1} & H^t_0 & H^t_1 & \dots & H^t_\theta & 0 & 0 & \dots \\ 0 & H^{t+1}_{-\tau} & \dots & H^{t+1}_{-1} & H^{t+1}_0 & H^{t+1}_1 & \dots & H^{t+1}_\theta & 0 & \dots \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & \dots & & & Q & & & \end{bmatrix} \begin{bmatrix} \Delta x_{t-\tau} \\ \vdots \\ \Delta x_{t-1} \\ \Delta x_t \\ \Delta x_{t+1} \\ \vdots \\ \Delta x_T \\ \Delta x_{T+1} \\ \vdots \\ \Delta x_{T+\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ F(x_t) - \epsilon_t \\ F(x_{t+1}) \\ \vdots \\ F(x_T) \\ Qx_+ \end{bmatrix}.$$

Note that it is equivalent to the original Newton step (equation 8) except for the t-period shock.

Performing a dynamic simulation using the $t-1$-period information set requires additional modification to our solution method. In this case, the expectations, $E(x_{t+})$, are formed prior to the realization of the shock, ϵ_t. We compute these using our usual method (as in section 1), with x_{t-} as initial conditions.

The period t "data" are generated by finding the x_t that solves the equation $F(x_{t-}, x_t, E(x_{t+})) = \epsilon_t$, taking the expectations and shock as given. To do this, we use a single-period Newton's method solution. The $t-1$ expectations are held fixed, inducing an $n\theta \times n\theta$ identity matrix in the bottom block of the Jacobian. There is one "stack" of H_i^t derivative matrices, since we are solving for x_t. The shock is fed into the system as a function residual as above. The full Newton step is, therefore,

$$\begin{bmatrix} & I_{n\tau \times n\tau} & & 0 & 0 & 0 & 0 \\ H_{-\tau}^t & \dots & H_{-1}^t & H_0^t & H_1^t & \dots & H_\theta^t \\ 0 & 0 & 0 & \dots & & I_{n\theta \times n\theta} & \end{bmatrix} \begin{bmatrix} \Delta x_{t-\tau} \\ \vdots \\ \Delta x_{t-1} \\ \Delta x_t \\ \Delta x_{t+1} \\ \vdots \\ \Delta x_{t+\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ F(x_t) - \epsilon_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

References

[1] Takeshi Amemiya. Non-linear regression models. In Zvi Griliches and Michael D. Intriligator, editors, *Handbook of Econometrics*, chapter 6, pages 333–89. North-Holland Publishing Company, Amsterdam, 1983.

[2] Gary Anderson and George Moore. A Linear Algebraic Procedure for Solving Linear Perfect Foresight Models. *Economics Letters*, 17:247–52, 1985.

[3] Gary Anderson and George Moore. An efficient procedure for solving nonlinear perfect foresight models. Working Paper, January 1986.

[4] Lawrence J. Christiano. Solving the stochastic growth model by linear-quadratic approximation and by value-function iteration. *Journal of Business and Economic Statistics*, 8(1):23–26, January 1990.

[5] Ray C. Fair. *Specification, Estimation, and Analysis of Macroeconometric Models*. Harvard University Press, Cambridge, MA, 1984.

[6] Ray C. Fair and John B. Taylor. Solution and maximum likelihood estimation of dynamic nonlinear rational expectations models. *Econometrica*, 51(4):1169-85, July 1983.

[7] Jeffrey C. Fuhrer and Brian R. Madigan. Monetary Policy When Interest Rates are Bounded at Zero. Federal Reserve Bank of Boston Working Paper #94–1, forthcoming *Review of Economics and Statistics*. May 1994.

[8] Jeffrey C. Fuhrer and George R. Moore. Inflation Persistence. *Quarterly Journal of Economics*, 110:127–59, February 1995a.

[9] Jeffrey C. Fuhrer and George Moore. Monetary policy trade-offs and the and the correlation between nominal interest rates and real output. *American Economic Review*, 85:219–39, March 1995b.

[10] Joseph E. Gagnon. Solving the stochastic growth model by deterministic extended path. *Journal of Business and Economic Statistics*, 8(1):35–36, January 1990.

[11] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, Inc., 1981.

[12] Robert G. King and Mark W. Watson. The Solution of Singular Linear Difference Systems Under Rational Expectations. Working Paper, April 1995a.

[13] Robert G. King and Mark W. Watson. System Reduction and Solution Algorithms for Singular Linear Difference Systems Under Rational Expectations. Working Paper, May 1995b.

[14] Finn E. Kydland and Edward C. Prescott. Time to build and aggregate fluctuations. *Econometrica*, 50(6):1345–1370, November 1982.

[15] Frederick R. Macaulay. *Some Theoretical Problems Suggested by the Movements of Interest Rates, Bond Yields, and Stock Prices in the United States Since 1856*. National Bureau of Economic Research, 1938.

[16] Bennett McCallum. Real business cycle models. In Robert J. Barro, editor, *Modern Business Cycle Theory*, chapter 1, pages 16–50. Harvard University Press, Cambridge, MA, 1989.

[17] Ellen R. McGrattan. Solving the stochastic growth model with a finite element method. *Journal of Economic Dynamics and Control*, 20(1–3):19–42, January 1996.

[18] Thomas J. Sargent. Estimation of Dynamic Labor Demand Schedules under Rational Expectations. *Journal of Political Economy*, 86, No. 6:1009–1044, December 1978.

[19] John B. Taylor and H. Uhlig. Solving nonlinear stochastic growth models: A comparison of alternative solution methods. *Journal of Business and Economic Statistics*, 8:1–17, January 1990.

Figure 1

Nonlinear Sticky Price Model: Unanticipated Disinflation

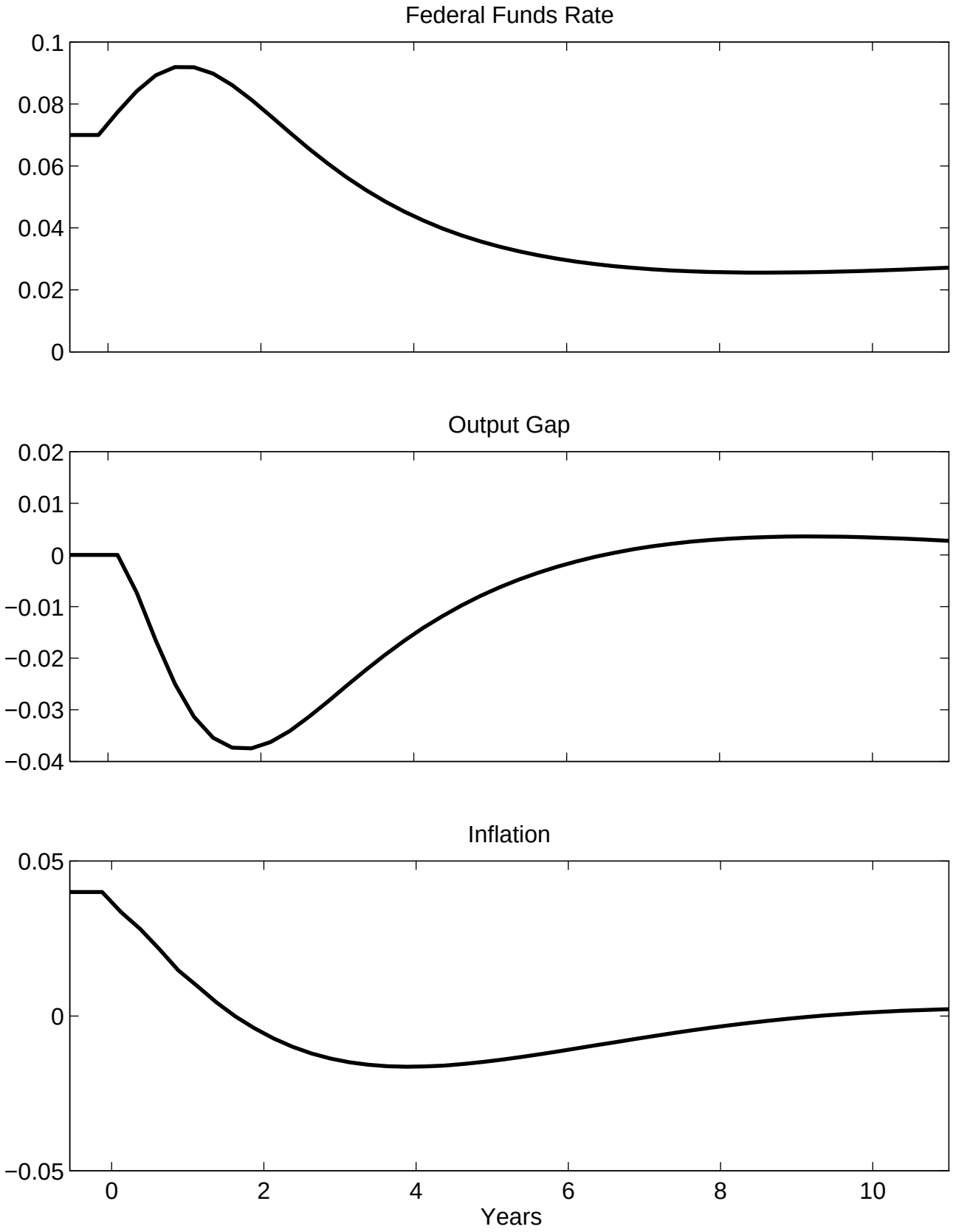

Figure 2

One−Sector Stochastic Growth Model: 10% Technology Shock to Steady State Levels

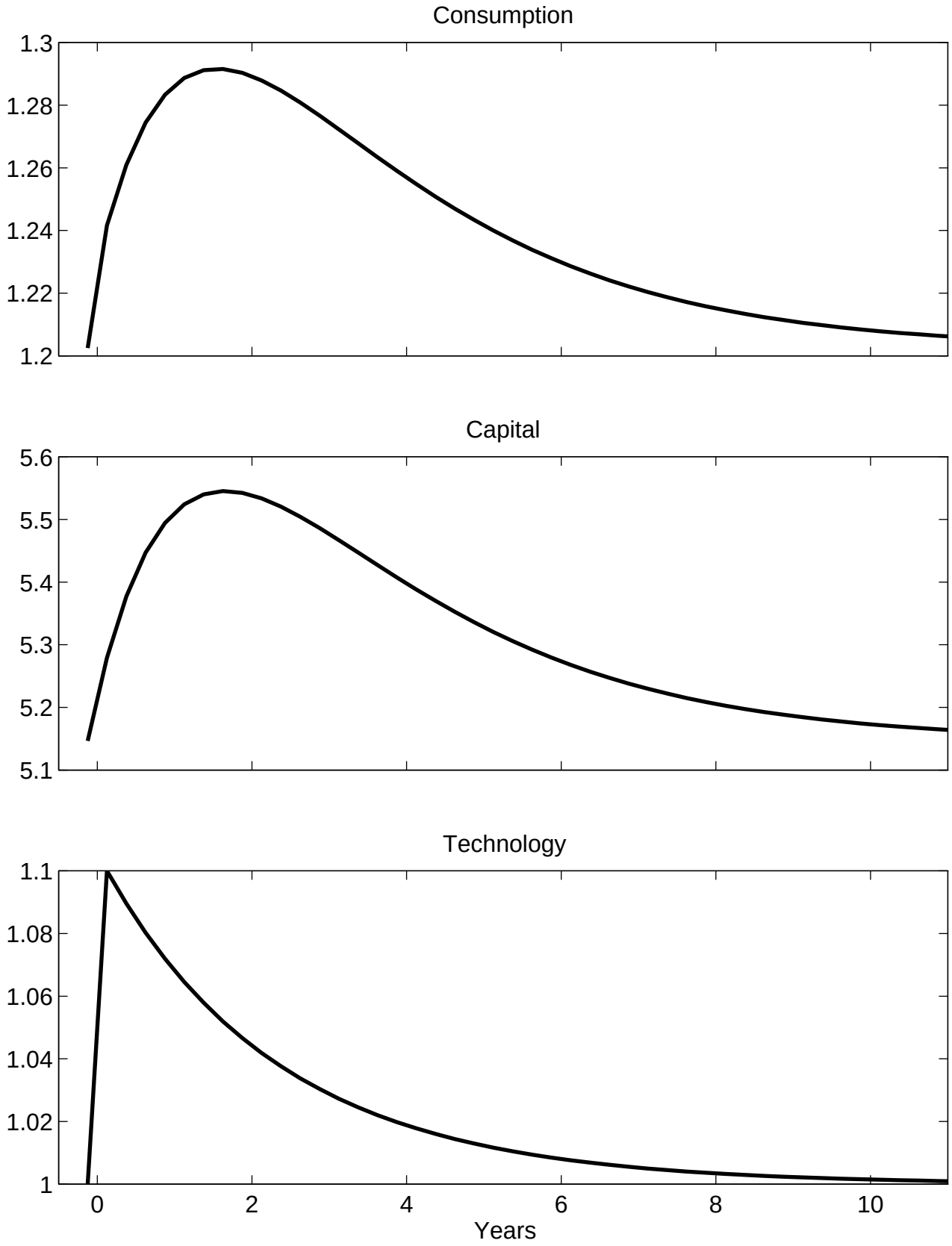

Figure 3

Two−Sector Stochastic Growth Model: 10% Technology Shock to Steady State Levels

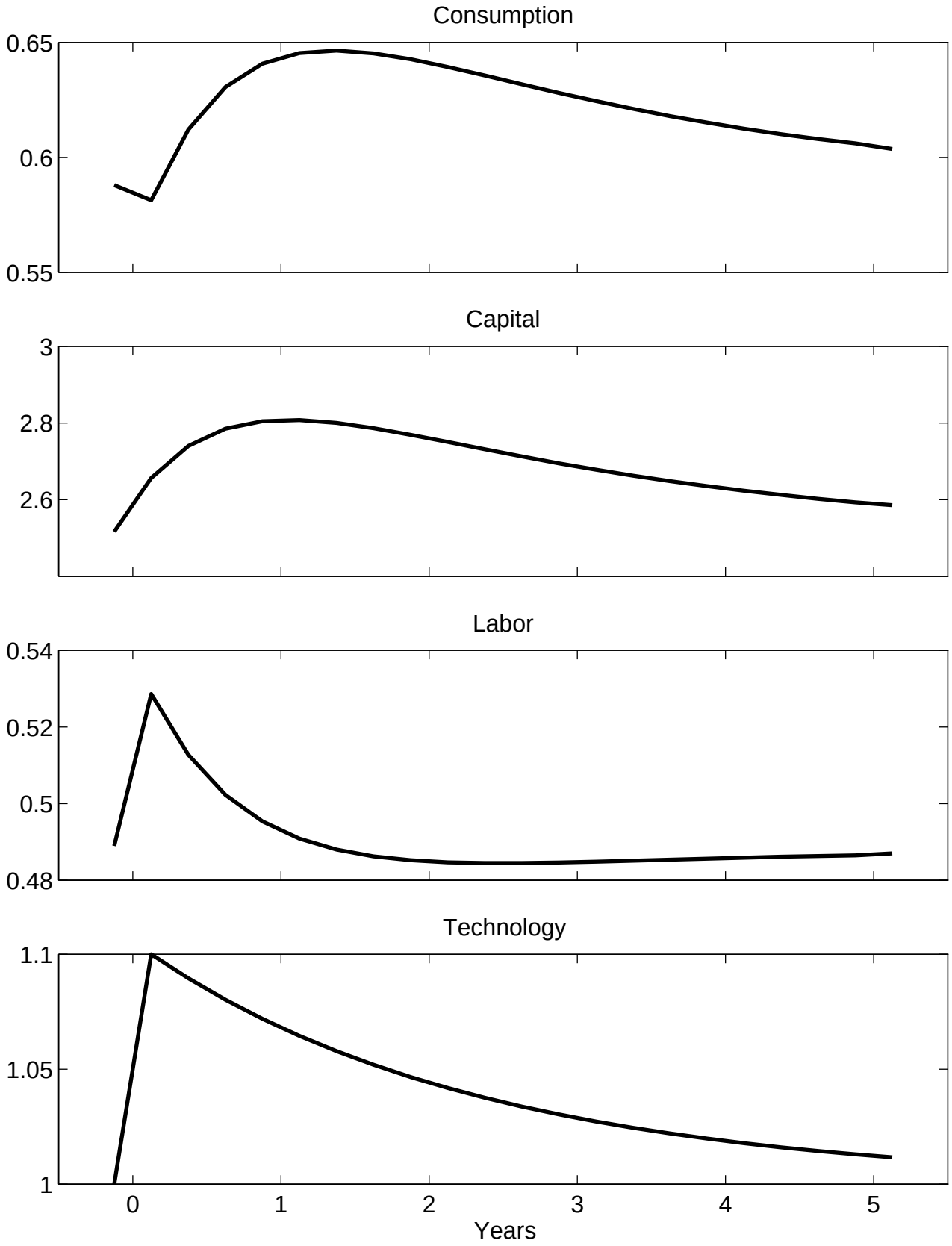

Figure 4

Our Method versus Dynamic Programming

Deterministic Solution Path

60% Initial Technology Shock

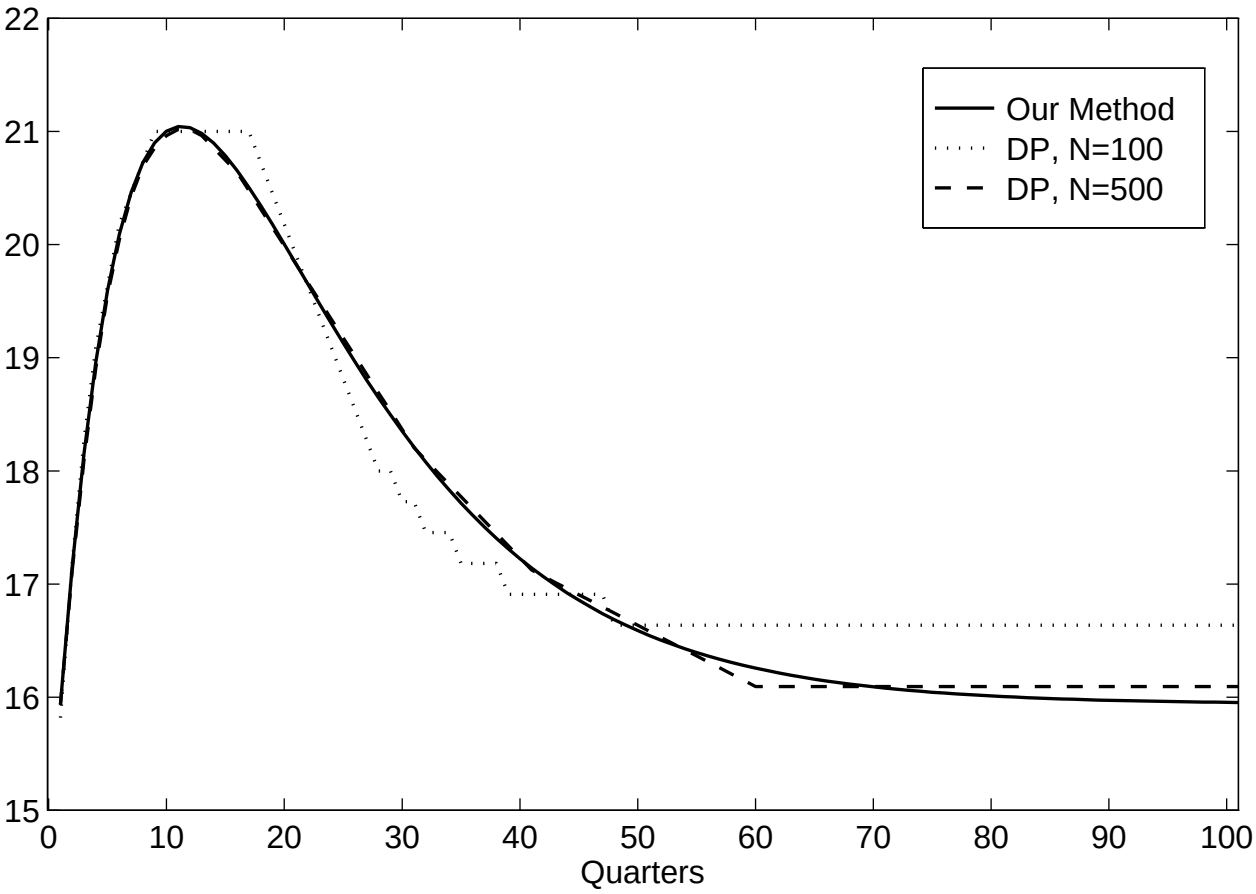

Figure 5

Consumption Decision Rules for the One−Sector Stochastic Growth Model: Finite Element versus Our Method